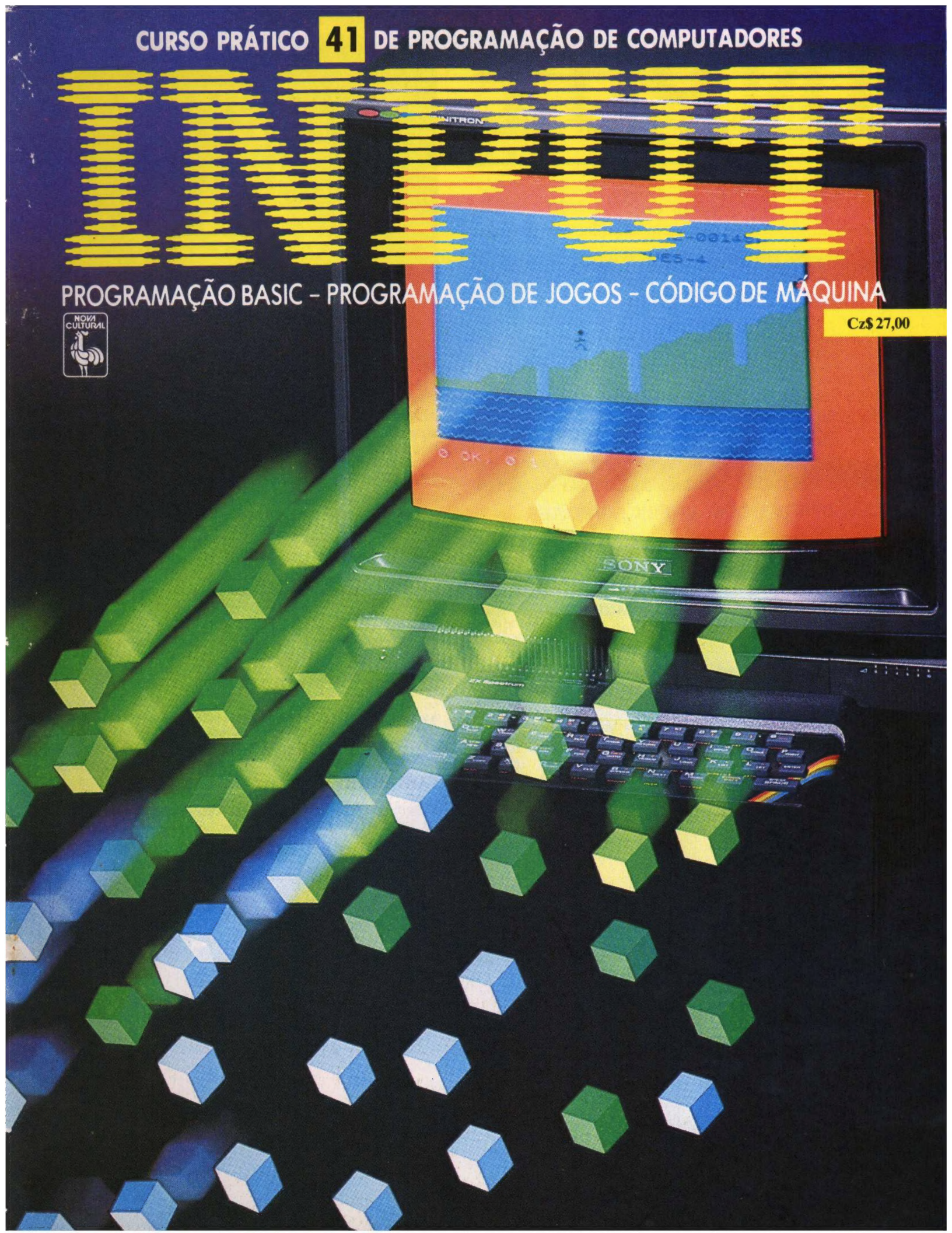
CURSO PRÁTICO 41 DE PROGRAMAÇÃO DE COMPUTADORES

TK 85

PROGRAMAÇÃO BASIC - PROGRAMAÇÃO DE JOGOS - CÓDIGO DE MÁQUINA

NOVA CULTURAL

Cz$ 27,00



INPUT

Vol. 3 N.º 41

NESTE NÚMERO

PROGRAMAÇÃO BASIC

FIGURAS GEOMÉTRICAS

Forma geométrica das mais fascinantes, o cone é responsável pela geração de toda uma família de curvas. Neste artigo apresentamos alguns programas para explorar as propriedades dessa figura e de suas seções 801

PROGRAMAÇÃO BASIC

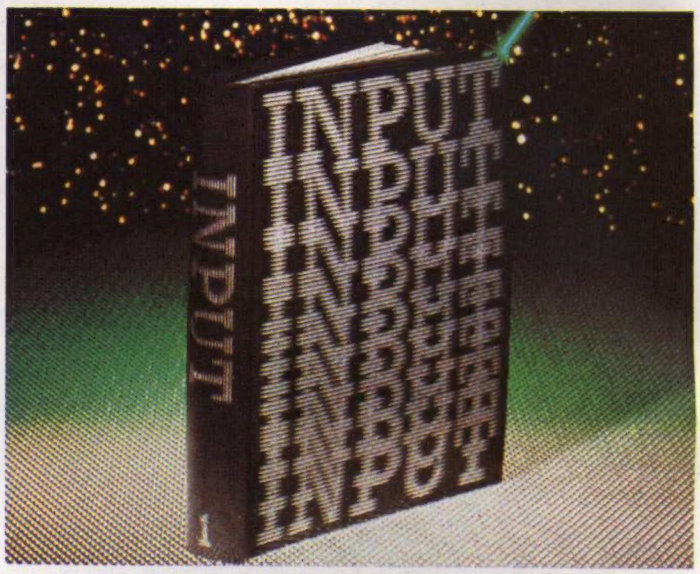
CRIE SPRITES COM VPEEK E VPOKE

Organização da memória de vídeo. Onde são armazenados os padrões dos sprites. Como o computador movimenta sprites. Um banco de sprites em linhas **DATA**. Um programa que ajuda a entender a organização dos sprites na VRAM. 808

CÓDIGO DE MÁQUINA

BLOCOS GRÁFICOS EM AVALANCHE

A procura do lanche roubado leva nosso indigitado herói a enfrentar novos obstáculos. Agora, ele precisa de uma montanha para escalar. Junte as porções do videogame. Defina a forma dos blocos gráficos. Mas procure não gravar bytes a mais. 816



PLANO DA OBRA

"INPUT" é uma obra editada em fascículos semanais, e cada conjunto de 15 fascículos compõe um volume. A capa para encadernação de cada volume estará à venda oportunamente.

COMPLETE SUA COLEÇÃO

Exemplares atrasados, até seis meses após o encerramento da coleção, poderão ser comprados, a preços atualizados, da seguinte forma: 1. PESSOALMENTE — Por meio de seu jornaleiro ou dirigindo-se ao distribuidor local, cujo endereço poderá ser facilmente conseguido junto a qualquer jornaleiro de sua cidade. Em **São Paulo**, os endereços são: rua Brigadeiro Tobias, 773, Centro; av. Industrial, 117, Santo André; e no **Rio de Janeiro**: rua da Passagem, 93, Botafogo. 2. POR CARTA — Poderão ser solicitados exemplares atrasados também por carta, que deve ser enviada para DINAP — Distribuidora Nacional de Publicações — Números Atrasados — Estrada Velha de Osasco, 132, Jardim Teresa — CEP 06000 — Osasco — SP. Não envie pagamento antecipado. O atendimento será feito pelo reembolso postal e o pagamento, incluindo as despesas postais, deverá ser efetuado ao se retirar a encomenda na agência do Correio. 3. POR TELEX — Utilize o nº (011) 33 670 DNAP.
Em **Portugal**, os pedidos devem ser feitos à Distribuidora Jardim de Publicações, Lda. — Qta. Pau Varais, Azinhaga de Fetais — 2 685, Camarate — Lisboa; Apartado 57 — Telex 43 069 JARLIS P.
Atenção: Após seis meses do encerramento da coleção, os pedidos serão atendidos dependendo da disponibilidade do estoque.
Obs.: Quando pedir livros, mencione sempre título e/ou autor da obra, além do número da edição.

COLABORE CONOSCO

Encaminhe seus comentários, críticas, sugestões ou reclamações ao Serviço de Atendimento ao Leitor — Caixa Postal 9442, São Paulo — SP.

Editor
VICTOR CIVITA

REDAÇÃO
Diretora Editorial: Iara Rodrigues

Editor Executivo: Antonio José Filho

Editor Chefe: Paulo de Almeida
Editor de Texto: Cláudio A. V. Cavalcanti
Chefe de Arte: Carlos Luiz Batista
Assistentes de Arte: Ailton Oliveira Lopes, Dilvacy M. Santos, Grace Alonso Arruda, José Maria de Oliveira, Monica Lenardon Corradi
Secretária de Redação/Coordenadora: Stefania Crema
Secretários de Redação: Beatriz Hagström, José Benedito de Oliveira Damião, Maria de Lourdes Carvalho, Marisa Soares de Andrade, Mauro de Queiroz

COLABORADORES
Consultor Editorial Responsável: Dr. Renato M. E. Sabbatini (Diretor do Núcleo de Informática Biomédica da Universidade Estadual de Campinas)
Execução Editorial: DATAQUEST Assessoria em Informática Ltda., Campinas, SP
Tradução: Reinaldo Cúrcio
Tradução, adaptação, programação e redação: Abílio Pedro Neto, Aluísio J. Dornellas de Barros, Marcelo R. Pires Therezo, Raul Neder Porrelli
Coordenação Geral: Rejane Felizatti Sabbatini
Editora de Texto: Ana Lúcia B. de Lucena
Assistente de Arte: Dagmar Bastos Sampaio

COMERCIAL
Diretor Comercial: Roberto Martins Silveira
Gerente Comercial: Flávio Maculan
Gerente de Circulação: Denise Maria Mozol

PRODUÇÃO
Gerente de Produção: João Stungis
Coordenador de Impressão: Atilio Roberto Bonon

Preparador de Texto/Coordenador: Eliel Silveira Cunha
Preparadores de Texto: Alzira Moreira Braz, Ana Maria Dilguerian, Karina Ap. V. Grechi, Levon Yacubian, Luciano Tasca, Maria Teresa Galluzzi, Maria Teresa Martins Lopes, Paulo Felipe Mendrone
Revisor/Coordenador: José Maria de Assis
Revisoras: Conceição Aparecida Gabriel, Isabel Leite de Camargo, Ligia Aparecida Ricetto, Maria de Fátima Cardoso, Nair Lúcia de Brito
Paste-up: Anastase Potaris, Balduino F. Leite, Edson Donato

© Marshall Cavendish Limited, 1984/85.
© Editora Nova Cultural Ltda., São Paulo, Brasil, 1986.

Edição organizada pela Editora Nova Cultural Ltda.
Av. Brigadeiro Faria Lima, nº 2000 - 3º andar
CEP 01452 - São Paulo - SP - Brasil
(Artigo 15 da Lei 5 988, de 14/12/1973).
Esta obra foi composta na AM Produções Gráficas Ltda.
e impressa na Divisão Gráfica da Editora Abril S.A.

FIGURAS GEOMÉTRICAS

- AS SEÇÕES DO CONE
- DESENHE CÍRCULOS, ELIPSES, PARÁBOLAS E HIPÉRBOLES
- ROTAÇÃO DE CURVAS
- APLICAÇÕES PRÁTICAS

Forma geométrica das mais fascinantes, o cone é responsável pela geração de toda uma família de curvas. Eis aqui alguns programas para explorar as propriedades dessa figura.

Criadores da geometria, os gregos antigos consideravam as curvas — pelas quais eram fascinados — tanto mais importantes quanto mais simples e elegantes eles fossem. Assim, quando se descobriu que toda uma família de curvas — conhecidas como seções cônicas — poderia ser obtida simplesmente cortando-se um cone de diferentes maneiras, pareceu óbvio que esses sólidos geométricos deviam ter uma significação especial. Realmente, dependendo de como se secciona o cone, pode-se obter um círculo, uma elipse, uma parábola ou uma hipérbole.

Na verdade, o traço marcante dessas curvas está no fato de que elas não são meras abstrações matemáticas, mas aparecem a todo momento no nosso cotidiano e proporcionam uma descrição exata de fenômenos físicos.

Existem, certamente, outras curvas simples encontradas na natureza que não são seções de um cone. A forma produzida por uma corda ou corrente presa por dois pontos nas extremidades é um exemplo. Conhecida como catenária, essa curva difere ligeiramente da parábola. Entretanto, as equações que servem para descrever as duas formas são completamente distintas. As seções cônicas aparecem freqüentemente relacionadas ao modo como as coisas se movem (um objeto lançado sobre a superfície da Terra, por exemplo, executa uma curva parabólica). Assim, elas são necessárias para uma simulação convincente e exata desses movimentos.

Algumas curvas também são úteis para definir objetos tridimensionais. Os cortes de um cone têm obviamente duas dimensões, mas podem ser rodados em seus eixos para produzir uma forma de três dimensões. Assim, o círculo se transforma em uma esfera, com um sem número de aplicações; a parábola, em um paraboloide, usado em objetos como faróis de automóveis, espelhos de telescópios, fornos solares etc.

A primeira parte deste artigo descreve cada curva e como desenhá-la na tela do computador, enquanto a segunda mostra como usá-la em simulações como a trajetória de um balde (ou outro objeto) pendurado em uma escada que desliza e acaba caindo (uma elipse) ou a curva descrita por uma pessoa cruzando um rio a nado (uma parábola).

Veremos também como desenhar belas formas usando a hipérbole e a elipse.

COMO CORTAR O CONE

As quatro curvas obtidas quando se seccionar um cone — o círculo, a elipse, a parábola e a hipérbole — diferem totalmente entre si. Foram estudadas em detalhe pela primeira vez pelo grego Apolônio, por volta de 200 a.C.

O ponto inicial é: se um par de retas que se cruzam — como um X — roda em torno de um eixo de simetria, gera-se um duplo cone (veja os desenhos destas páginas) que pode ser cortado por um plano de quatro maneiras.

Se um corte é feito em ângulo reto ao eixo de simetria, a seção resultante é um círculo.

Um corte feito em um ângulo entre 90° e metade do ângulo formado pelas linhas que geraram o cone (conhecido como ângulo semivertical do cone) produz uma elipse.

Um plano que faz com o eixo um ângulo igual ao ângulo semivertical nos dá uma parábola. Já um plano que faz com o eixo um ângulo menor que o semivertical cria uma seção em duas partes chamada hipérbole. Ela tem duas partes porque nosso plano corta tanto o cone de cima como o de baixo (lembre-se de que o duplo cone foi gerado pela rotação de um X).

Existem dois casos especiais. Se, por exemplo, o corte é feito por um plano que passa pelo eixo, isto é, se os cones são divididos verticalmente pela metade, serão obtidas duas linhas retas — aquelas que foram usadas para gerar a figura. Elas configuram na realidade um caso especial de hipérbole. Por outro lado, se um plano cortar os cones pelo vértice, fazendo um ângulo reto com o eixo vertical, tudo o que se tem é um ponto, que é um círculo de raio zero.

Os desenhos destas páginas devem deixar claro como tais formas são obtidas. Se quiser, você também pode cortar cones feitos de material de fácil manipulação — como isopor ou papel — e verificar o que obtém. Um cone duplo só será necessário se você quiser ter uma hipérbole de verdade, visto que ela sempre tem duas partes.

CURVAS E MAIS CURVAS

As curvas são geradas por equações simples, algumas das quais já foram abordadas em outros programas. Elas serão sempre desenhadas na tela de alta resolução do seu micro. Vejamos agora algumas delas. Comecemos pelo círculo, uma das curvas mais simples.

O CÍRCULO

A equação de um círculo é dada por:

X = A*COS teta
Y = A*SEN teta

onde A é o raio. X e Y, um ponto qualquer da circunferência, e teta, o ângulo formado com uma reta prefixada, em geral o eixo X.

O primeiro programa desenha um círculo de raio A no meio da tela:

```
10 CLS
15 LET a=70
25 LET x=a: LET y=0
30 PLOT 127+x,70+y
40 FOR t=0 TO 2*PI STEP .2
50 LET x=a*COS t: LET y=a*SIN
 t
60 DRAW x-PEEK 23677+127,y-
PEEK 23678+70
70 NEXT t
```

```
10 PMODE 4:PCLS:SCREEN 1,1
15 A=60
20 C=ATN(1)/45
30 LINE-(127+A,95),PRESET
40 FOR TH=0 TO 360 STEP 10
50 X=A*COS(TH*C):Y=A*SIN(TH*C)
60 LINE -(127+X,95+Y),PSET
70 NEXT TH
80 GOTO 80
```

```
10 COLOR 15,4,4:SCREEN2
15 A=60
20 C=ATN(1)/45
30 LINE -(127+A,95),4
40 FOR TH=0 TO 360 STEP 10
50 X=A*COS(TH*C):Y=A*SIN(TH*C)
60 LINE -(127+X,95+Y),15
70 NEXT
80 GOTO 80
```

```
10  HOME : HGR : HCOLOR= 3
15 A = 60
20 C =  ATN (1) / 45
30  HPLOT 127 + A,95
40  FOR TH = 0 TO 360 STEP 10
50 X = A *  COS (TH * C):Y = A
*  SIN (TH * C)
60  HPLOT  TO 127 + X,95 + Y
70  NEXT
```

O laço **FOR...NEXT** das linhas 40 a 70 é a parte do programa que desenha o círculo, traçando repetidamente segmentos de reta a intervalos de 10° (ou 0.2 radianos). O raio do círculo é definido na linha 15.

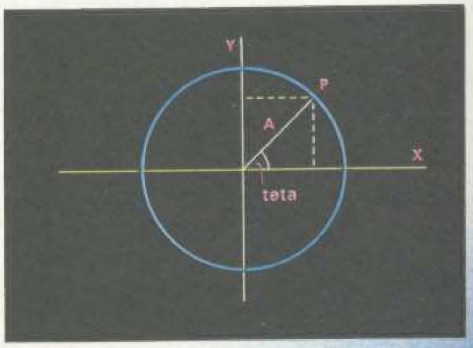
A ELIPSE

A equação da elipse é muito parecida com a do círculo. Para uma elipse com o eixo maior 2A e o menor 2B, a posição do ponto da periferia é:

X = A*COS teta
Y = B*SEN teta

A forma da elipse — ou seja, suas proporções de características mais ou menos achatadas — é determinada por A e B. Altere estas linhas:

```
16 LET b=40
50 LET x=a*COS t: LET y=b*SIN
t
```

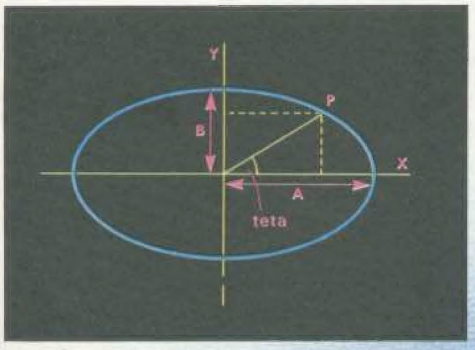


O círculo

```
16 B=30
50 X=A*COS(TH*C):Y=B*SIN(TH*C)
```

```
16 B=30
50 X=A*COS(TH*C):Y=B*SIN(TH*C)
```

```
16 B = 30
50 X = A *  COS (TH * C):Y = B
*  SIN (TH * C)
```

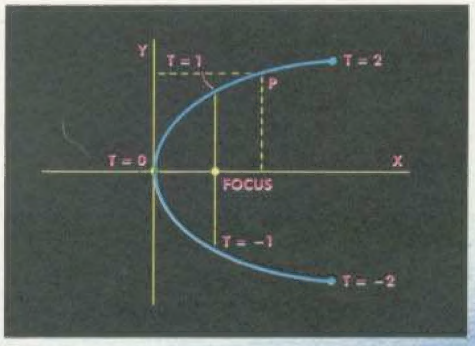


A elipse

A PARÁBOLA

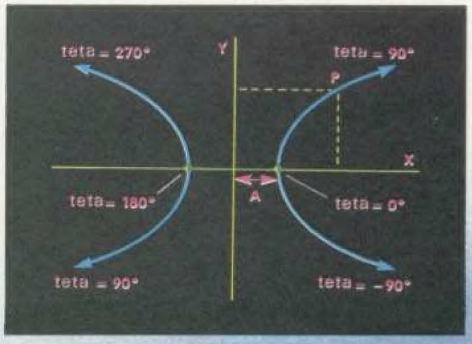
O tamanho da parábola depende do valor de uma variável, conhecida por T. As equações são:

X = T
Y = 2*T



A parábola

O valor de T pode variar de infinito a menos infinito; grande parte de uma parábola, contudo, pode ser vista como T variando entre +2 e −2. No programa, esses valores têm que ser postos em escala por um fator M para que a curva caiba na tela do micro.



A hipérbole

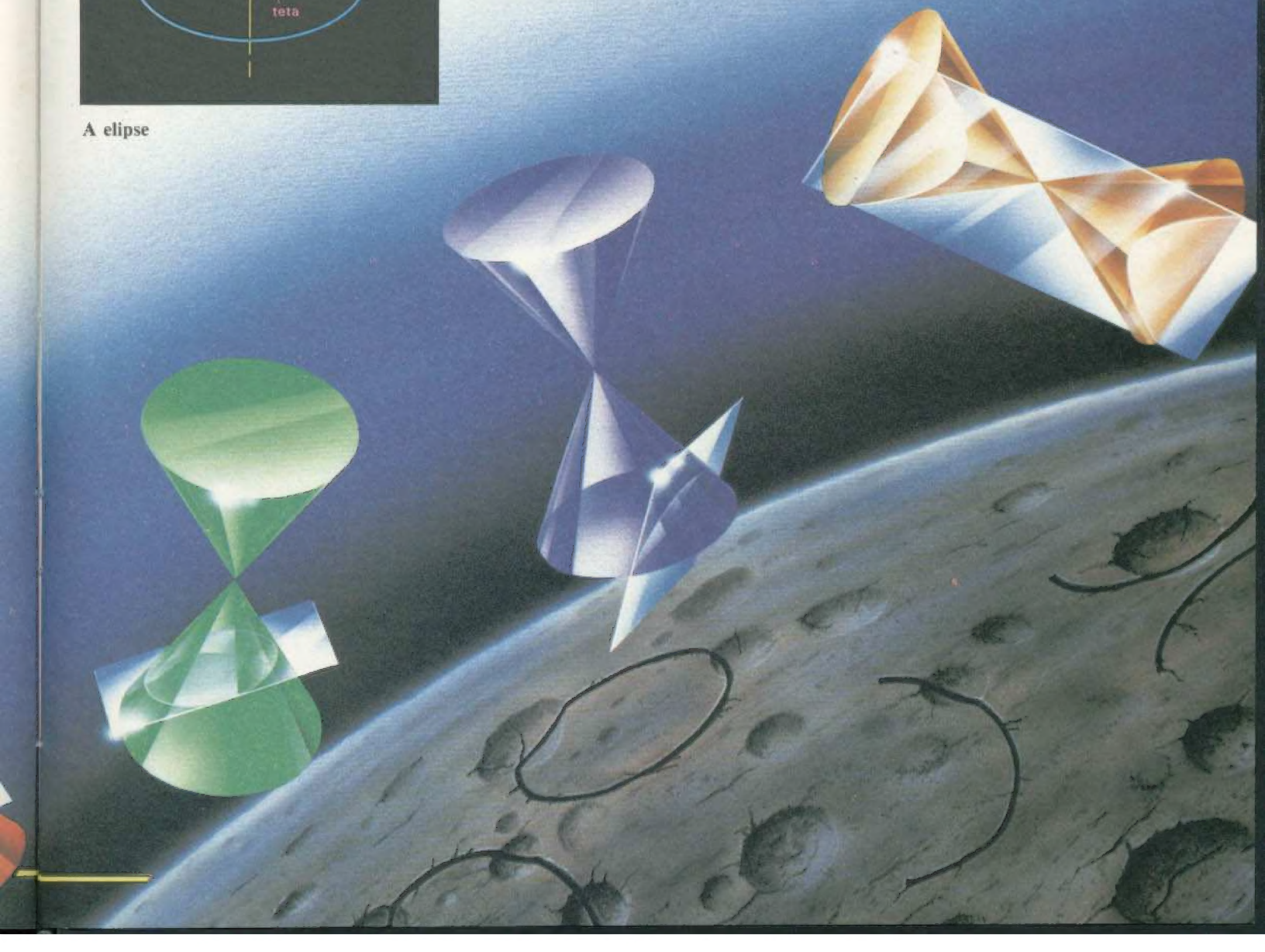
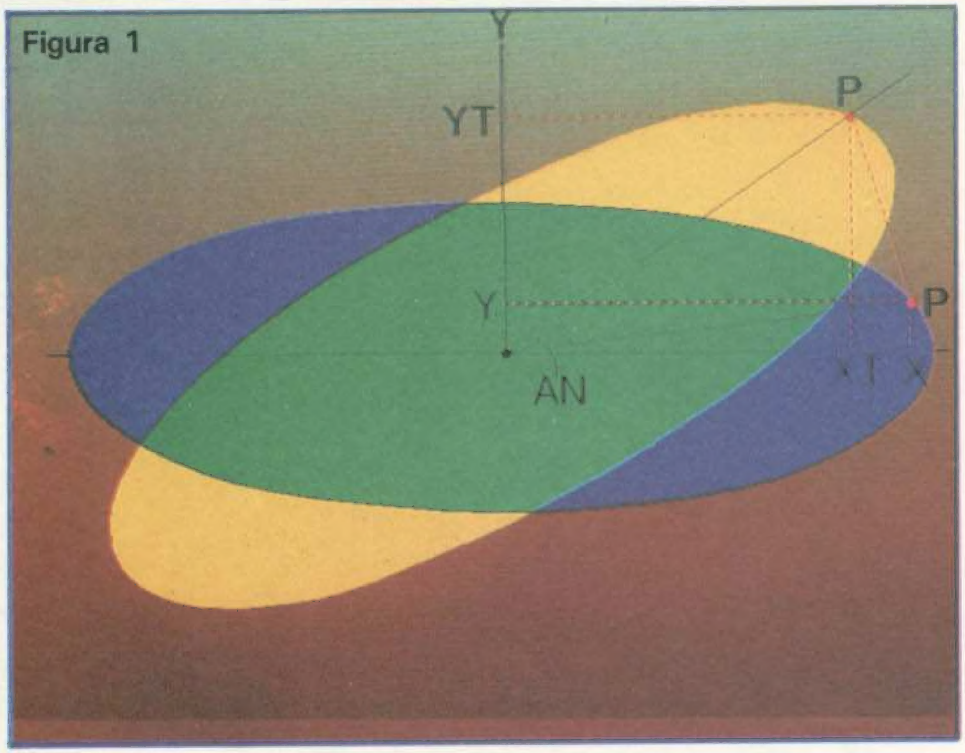


Figura 1




```
10 CLS
15 LET m=20
25 LET x=4*m: LET y=-4*m
30 PLOT 127+x,80+y
40 FOR t=-2 TO 2 STEP .05
50 LET x=m*t*t: LET y=2*m*t
60 DRAW x-PEEK 23677+127,y-
PEEK 23678+80
70 NEXT t
```

```
10 PMODE 4:PCLS:SCREEN 1,1
15 M=23
20 C=ATN(1)/45
30 LINE-(127+M*4,95-4*M),PRESET
40 FOR T=-2 TO 2 STEP .05
50 X=M*T^2:Y=2*M*T
60 LINE-(127+X,95+Y),PSET
70 NEXT T
80 GOTO 80
```

```
10 COLOR15,4,4:SCREEN2
15 M=23
20 C=ATN(1)/45
30 LINE -(127+M*4,95-M*4),4
40 FOR T=-2 TO 2 STEP .05
50 X=M*T^2:Y=2*M*T
60 LINE -(127+X,95+Y),15
70 NEXT
80 GOTO 80
```

```
10  HOME : HGR2 : HCOLOR= 3
15 M = 20
20 C =  ATN (1) / 45
30  HPLOT 127 + M * 4,95 - 4 *
M
40  FOR T =  - 2 TO 2 STEP .05
50 X = M * T ^ 2:Y = 2 * M * T
60  HPLOT  TO 127 + X,95 + Y
70  NEXT
```

A HIPÉRBOLE

A equação da hipérbole é:

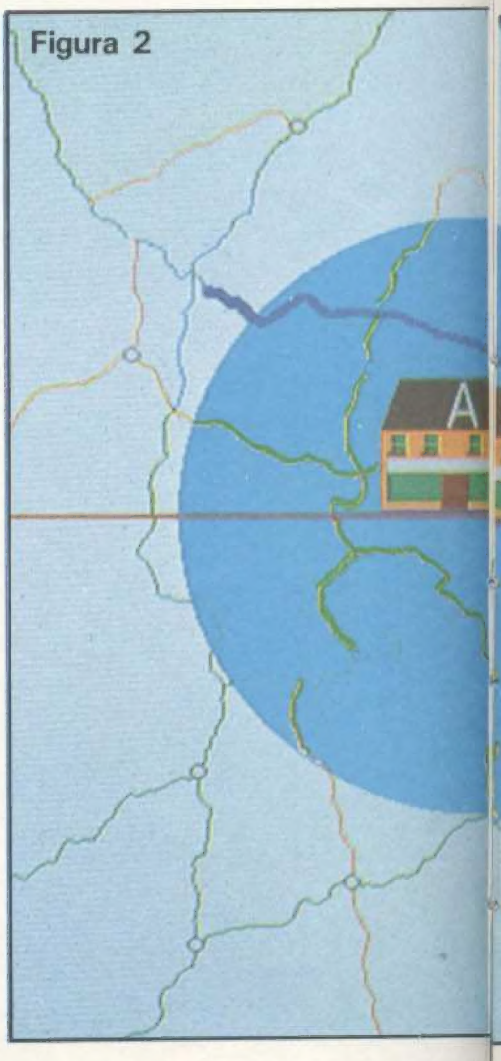
X = A/COS teta
Y = B*TG teta

Metade da hipérbole é traçada enquanto teta varia de -90° a 90° e a outra metade, enquanto teta varia de 90° a 270°. Teoricamente, é possível usar apenas um laço no programa para variar teta de -90 a +270, mas há problemas nos pontos -90, 90 e 270, onde ocorre uma divisão por zero. Mesmo com valores próximos destes, grandes números são envolvidos. Assim, o nosso programa recorre a dois laços. Novamente, o fator M vem estabelecer a escala mais adequada para a figura:

```
10 CLS
15 LET m=30
25 LET x=m/COS -1: LET y=m*
TAN -1
30 PLOT 127+x,75+y
40 FOR t=-1 TO 1 STEP .1
50 LET x=m/COS t: LET y=m*TAN
 t
60 DRAW 127+x-PEEK 23677,75+y
-PEEK 23678
70 NEXT t
75 LET x=m/COS (PI-1): LET y=
m*TAN (PI-1)
80 PLOT 127+x,75+y
90 FOR t=PI-1 TO PI+1 STEP .1
100 LET x=m/COS t: LET y=m*TAN
 t
110 DRAW 127+x-PEEK 23677,75+y
-PEEK 23678
120 NEXT t
```

```
10 PMODE 4:PCLS:SCREEN 1,1
15 M=50
20 C=ATN(1)/45
30 LINE -(227,8),PRESET
40 FOR TH=-60 TO 60 STEP 5
50 X=M/COS(TH*C):Y=M*TAN(TH*C)
60 LINE -(127+X,95+Y),PSET
70 NEXT TH
80 LINE-(26,8),PRESET
90 FOR TH=120 TO 240 STEP 5
100 X=M/COS(TH*C):Y=M*TAN(TH*C)
110 LINE-(127+X,95+Y),PSET
120 NEXT TH
130 GOTO 130
```

Figura 2



```
10 COLOR15,4,4:SCREEN2
15 M=50
20 C=ATN(1)/45
30 LINE -(227,8),4
40 FOR TH=-60 TO 60 STEP 5
50 X=M/COS(TH*C):Y=M*TAN(TH*C)
60 LINE -P127+X,95+Y),15
70 NEXT
80 LINE -(26,8),4
90 FOR TH=120 TO 240 STEP 5
100 X=M/COS(TH*C):Y=M*TAN(TH*C)
110 LINE -(127+X,95+Y),15
120 NEXT
130 GOTO 130
```

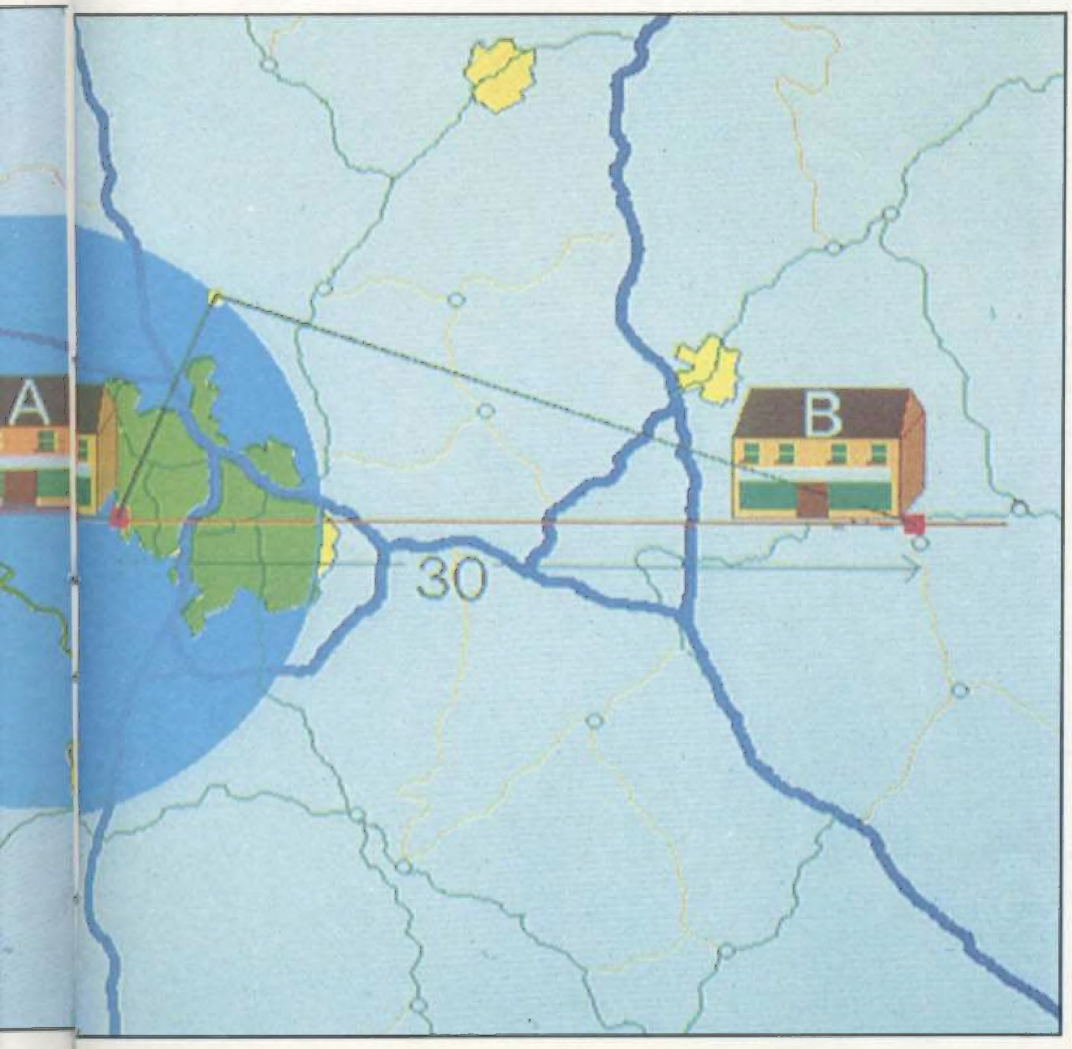
```
10  HOME : HGR2 : HCOLOR= 3
15 M = 50
20 C =  ATN (1) / 45
30  HPLOT 227,8
40  FOR TH =  - 60 TO 60 STEP 5

50 X = M /  COS (TH * C):Y = M
*  TAN (TH * C)
60  HPLOT  TO 127 + X,95 + Y
70  NEXT
80  HPLOT 26,8
90  FOR TH = 120 TO 240 STEP 5
100 X = M /  COS (TH * C):Y = M
 *  TAN (TH * C)
110  HPLOT  TO 127 + X,95 + Y
120  NEXT
```

COMO RODAR AS CURVAS

Os programas anteriores desenharam as curvas na sua forma mais simples, sobre um eixo horizontal e outro vertical. No entanto, essa posição nem sempre é conveniente; além disso, pode-se querer desenhar as curvas em outra posição. A figura 1 mostra o que acontece com um ponto na periferia de uma elipse quando esta sofre uma rotação de um ângulo de AN graus. O ponto P move-se da posição X,Y para uma nova posição XT, YT e as novas coordenadas são dadas por:

XT = X*COS AN-Y*SEN AN
YT = X*SEN AN + Y*COS AN



Eis aqui a rotina de rotação para cada computador:

```
1000 LET xt=x*COS (an*PI/180)-y
*SIN (an*PI/180)
1010 LET yt=x*SIN (an*PI/180)+y
*COS (an*PI/180)
1020 RETURN
```

```
1000 XT=X*COS(AN*C)-Y*SIN(AN*C)
1010 YT=Y*COS(AN*C)+X*SIN(AN*C)
1020 RETURN
```

Algumas alterações deverão ser feitas nos programas que desenham as curvas para que eles possam usar a subrotina de rotação. Em primeiro lugar, o ângulo de rotação tem que ser especificado (linha 17); a posição inicial deve ser rodada e as linhas, desenhadas nas novas coordenadas XT e YT. A linha 17 pode ser alterada para permitir a entrada do valor do ângulo para rotação por meio de uma instrução **INPUT**. É importante lembrar que essa instrução deve ser colocada antes da seleção do modo gráfico.

Vejamos agora quais são as mudanças que precisamos realizar no programa BASIC que desenha a elipse. Não se esqueça de incorporar a rotina de rotação a cada programa.

```
17 LET an=60
28 GOSUB 1000
30 PLOT 127+xt,70+yt
55 GOSUB 1000
60 DRAW xt-PEEK 23677+127,yt-
PEEK 23678+70
80 STOP
```

```
17 AN=60
25 X=A:GOSUB 1000
30 LINE-(127+XT,95+YT),PRESET
55 GOSUB 1000
60 LINE-(127+XT,95+YT),PSET
```

```
17 AN=60
25 X=A:GOSUB 1000
30 LINE -(127+XT,95+YT),4
55 GOSUB 1000
60 LINE -(XT+127,YT+95),15
```

```
17 AN = 60
25 X = A: GOSUB 1000
30  HPLOT 127 + XT,95 + YT
55  GOSUB 1000
60  HPLOT  TO 127 + XT,95 + YT
80  END
```

Para fazer a rotação da parábola deve-se empregar a mesma sub-rotina. Adicione-a ao programa principal e faça as seguintes alterações:

```
17 LET an=60
28 GOSUB 1000
30 PLOT 127+xt,80+yt
40 FOR t=-1.75 TO 1.75 STEP
.05
55 GOSUB 1000
60 DRAW 127+xt-PEEK 23677,80+
yt-PEEK 23678
80 STOP
```

```
17 AN=60
20 C=ATN(1)/45
25 X=M*4:Y=-M*4:GOSUB 1000
30 LINE -(127+XT,95+YT),PRESET
55 GOSUB 1000
60 LINE -(127+XT,95+YT),PSET
```

```
17 AN=60
25 X=M*4:Y=-M*4:GOSUB 1000
30 LINE -(127+XT,95+YT),4
55 GOSUB 1000
60 LINE -(XT+127,YT+95),15
```

```
15 M = 17
17 AN = 60
20 C =  ATN (1) / 45
25 X = M * 4:Y =  - M * 4: GOSU
B 1000
30  HPLOT 127 + XT,95 + YT
55  GOSUB 1000
60  HPLOT  TO 127 + XT,95 + YT
80  END
```

E, finalmente, as alterações para rodar a hipérbole:

```
 17 LET an=60
 28 GOSUB 1000
 30 PLOT 127+xt,75+yt
 55 GOSUB 1000
 60 DRAW 127+xt-PEEK 23677,75+
 yt-PEEK 23678
 76 GOSUB 1000
 80 PLOT 127+xt,75+yt
105 GOSUB 1000
110 DRAW 127+xt-PEEK 23677,75+
yt-PEEK 23678
130 STOP
```

```
17 AN=60
25 X=M/COS(-60*C):Y=M*TAN(-60*C
):GOSUB 1000
30 LINE -(127+XT,95+YT),PRESET
55 GOSUB 1000
60 LINE -(XT+127,YT+95),PSET
75 X=M/COS(135*C):Y=M*TAN(135*C
):GOSUB 1000
80 DRAW"BM"+STR$(INT(127+XT))+"
,"+STR$(INT(95+YT))
90 FOR TH=135 TO 240 STEP 5
105 GOSUB 1000
110 LINE -(127+XT,95+YT),PSET
```

```
17 AN=60
25 X=M/COS(-60*C):Y=M*TAN(-60*C
):GOSUB 1000
30 LINE -(127+XT,95+YT),4
55 GOSUB 1000
60 LINE -(XT+127,YT+95),15
75 X=M/COS(135*C):Y=M*TAN(135*C
):GOSUB 1000
80 LINE -(127+XY,95+YT),4
90 FOR TH=135 TO 240 STEP 5
105 GOSUB 1000
110 LINE -(XT+127,YT+95),15
```

```
15 M = 35
17 AN = 60
25 X = M /  COS ( - 60 * C):Y =
 M *  TAN ( - 60 * C): GOSUB 10
```

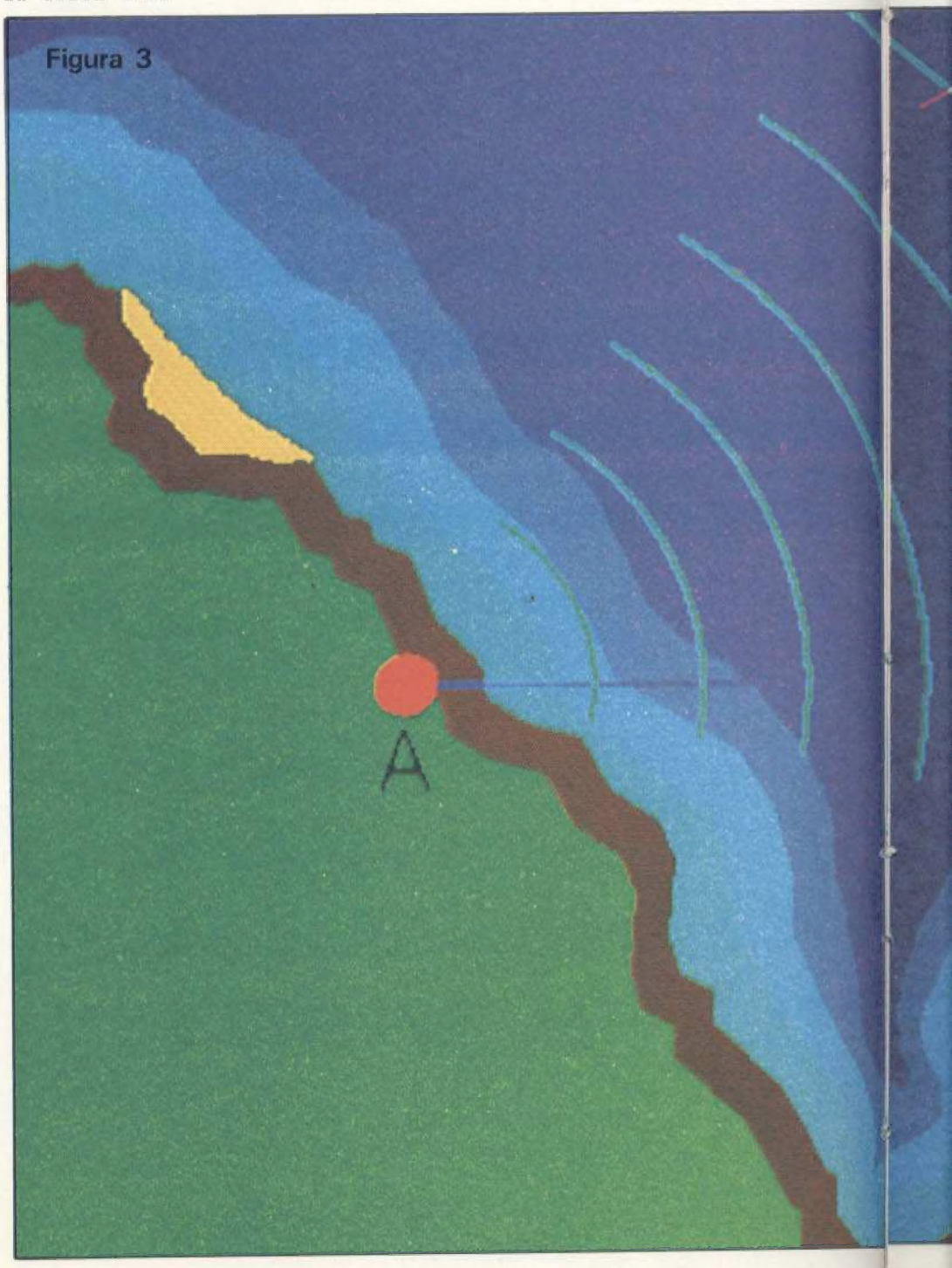
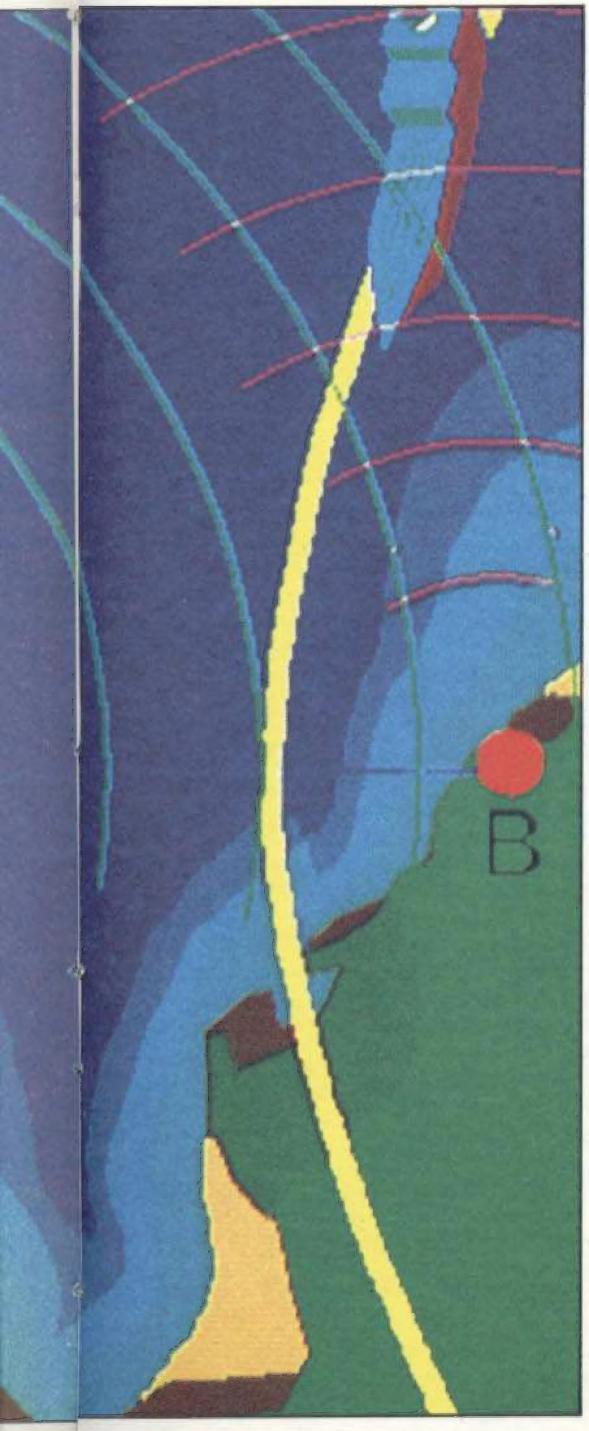


Figura 3

```
00
30  HPLOT 127 + XT,95 + YT
55  GOSUB 1000
60  HPLOT  TO 127 + XT,95 + YT
75 X = M /  COS (135 * C):Y = M
 *  TAN (135 * C): GOSUB 1000
80  HPLOT 127 + XT,95 + YT
90  FOR TH = 135 TO 240 STEP 5
105  GOSUB 1000
110  HPLOT  TO 127 + XT,95 + YT

130  END
```



APLICAÇÕES PRÁTICAS

Todas essas curvas podem ser usadas de alguma forma prática.

O círculo tem tantas aplicações que nem todas podem ser listadas. A roda é um exemplo óbvio, assim como a bola também o é para a esfera. As esferas (ou formas aproximadas) aparecem com freqüência na natureza. Os exemplos vão desde laranjas até planetas. Mas elas raramente são perfeitas, devido à ação da gravidade, ventos ou outras forças. Um planeta rodando ao redor de uma estrela poderia ter uma órbita circular, apesar de ser mais provável que ela fosse elíptica.

As aplicações para o círculo são tão amplas como a determinação do menor custo de transporte para um produto que pode ser comprado em dois lugares diferentes. Suponhamos, por exemplo, que você queira comprar um modelo de computador vendido pelas firmas A e B, que estão distante 300 km entre si. Suponhamos ainda que a firma A envie o computador por um serviço especial de entrega a um custo de um cruzado por km, enquanto a firma B o faça em seu próprio caminhão, ao custo de cinqüenta centavos por km. É muito simples marcar num mapa a região em que é mais barato comprar de A ou de B. A idéia é unir por meio de uma linha todos os pontos onde os custos são iguais. Neste caso, devemos definir os lugares em que a distância até B seja o dobro da distância até A.

Um desses pontos fica na linha que liga as duas firmas, a 100 km de A e 200 km de B. Outro ponto fica na mesma linha, a 300 km de A, no sentido oposto a B. Se todos esses pontos forem unidos, teremos um círculo de raio igual a 200 km, como se vê na figura 2. Se você vive dentro do círculo, é mais barato comprar de A; se vive fora, é mais compensador comprar de B.

A elipse também tem aplicações práticas. É possível, por exemplo, colocar uma elipse em tal posição que sua sombra, projetada sobre uma superfície plana, assuma a forma de um círculo perfeito. Essa propriedade pode ser usada em válvulas de dutos circulares, onde uma aba elíptica é usada para controlar o fluxo de líquidos ou gases. Ao atingir determinado ângulo, a aba se encaixa perfeitamente no duto, bloqueando o fluxo da substância. A parábola descreve a curva traçada por um projétil. Mas certos corpos celestes, como os cometas, podem também viajar em órbitas parabólicas em torno do Sol e de outras estrelas.

MICRO DICAS

PARÁBOLAS E HIPÉRBOLES

Os programas deste artigo foram concebidos de modo a explorar da melhor maneira a tela do seu micro. Ao usar essas formas em programas, o fator de escala M deve ser alterado para que sejam obtidas curvas do tamanho adequado.

Ao rodar parábolas e hipérboles, é necessário tomar certos cuidados para que os pontos das extremidades das curvas não caiam fora da tela. Assim, altere o fim do laço **FOR...NEXT** da linha 40 do programa da parábola e das linhas 40 a 90 do programa da hipérbole. O limite exato deverá ser encontrado pelo método de tentativa e erro.

Por outro lado, raios de luz e calor paralelos ao eixo de uma parábola são refletidos de modo a passar pelos focos. Isso acontece nas duas direções. Assim, um filamento colocado no foco produzirá um feixe paralelo de luz, como os usados em faróis de automóveis. Na outra direção, os raios solares podem ser concentrados no foco de uma parábola, produzindo altíssimas temperaturas; este é o princípio de funcionamento dos fornos solares.

Na prática, os refletores utilizados para tais fins são paraboloides tridimensionais. Estes também são usados em outras aplicações, como antenas para recepção e transmissão de ondas de rádio, acessórios de sistemas de televisão e telefonia etc.

Uma característica importante da hipérbole é — como já foi dito — sua constituição em duas partes. Os sistemas de radar para navegação valem-se dessa qualidade, funcionando com duas estações. Uma delas transmite um sinal, que é retransmitido imediatamente pela outra estação. Ao receber os dois sinais, o operador de um navio nas proximidades calcula o tempo entre as duas emissões. Se o barco se move de maneira a manter constante o tempo entre as chegadas, é porque tem um curso hiperbólico, como é mostrado na figura 3. Se o operador também receber sinais de outras duas estações e calcular o tempo entre a chegada dos sinais, terá outra hipérbole; a interseção das duas dará a posição do navio.

CRIE SPRITES COM VPEEK E VPOKE

Você pode criar e movimentar sprites com a ajuda dos versáteis comandos VPOKE e VPEEK. Mas, para isso, precisa conhecer primeiro a organização de memória de vídeo do MSX.

Até agora, usamos bastante os comandos **VPEEK** e **VPOKE** para obter efeitos gráficos, mas nunca na programação de sprites. Para fazer isso, precisamos conhecer bem a organização da memória de vídeo do MSX, a chamada VRAM. A estrutura dessa memória, bem como os princípios básicos dos comandos **VPEEK** e **VPOKE**, foi explicada no artigo *Os Comandos PEEK e POKE* (página 261).

A VRAM é uma memória independente, com capacidade de armazenar até 16 kbytes. Ela é utilizada por um *chip* especial, que se dedica integralmente ao controle da tela: o VDP — *Video Display Processor*.

O MSX pode dispor da tela de quatro formas: texto em quarenta colunas, texto em 32 colunas, gráficos em alta resolução e gráficos em baixa resolução. É o VDP que determina o tipo de tela que está sendo mostrado no vídeo. Quando estamos programando em BASIC, o comando **SCREEN** ajusta o VDP para mostrar o tipo de tela desejado.

Os diferentes tipos de tela nada mais são que maneiras diversas de interpretar os números que estão na VRAM. Para fazer essa interpretação, o VDP divide a memória de vídeo em regiões que são denominadas tabelas. Cada tabela tem sua função. No modo texto de quarenta colunas, por exemplo, existe uma tabela de nomes, cuja função consiste em armazenar os códigos ASCII das letras que aparecem no texto. Uma vez que coloquemos um código numa certa posição da tabela de nomes, o VDP utiliza esse valor a fim de encontrar os bytes que determinam o formato da letra em outra parte da VRAM (na chamada tabela de padrões).

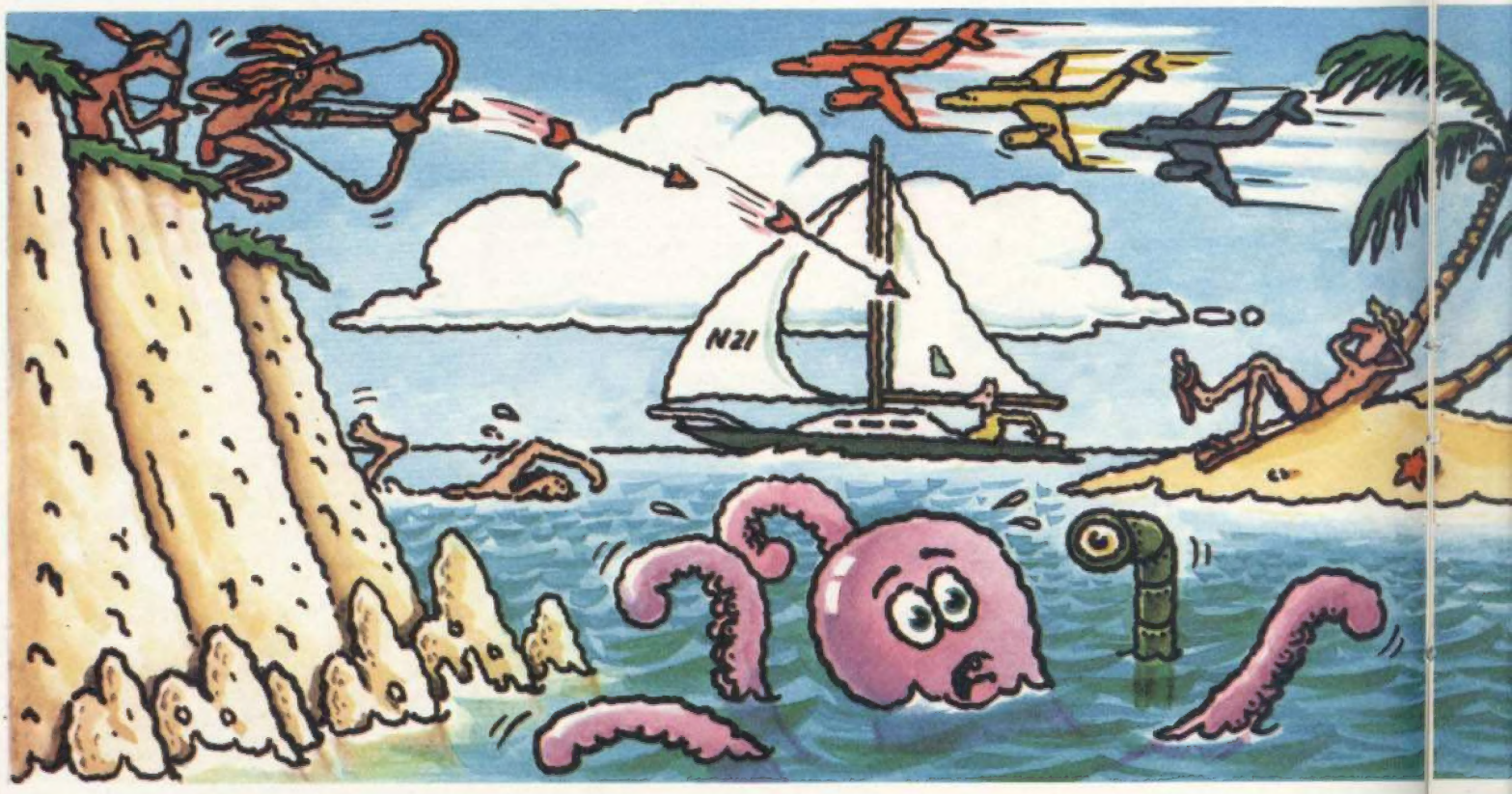
Os quatro tipos de tela têm tabelas de nomes e tabelas de padrões, que funcionam de maneira parecida. Para que não seja preciso memorizar os endereços iniciais de cada uma delas na VRAM, empregamos variáveis do sistema chamadas **BASE**, que descobrem o valor correto para nós. Esses valores estão contidos no VDP; em outro artigo aprenderemos a modificá-los.

No modo texto de quarenta colunas — **SCREEN 1** — não podemos modificar a cor de letras individuais nem usar sprites. Nos outros três tipos de tela existe uma porção da VRAM, dedicada a controlar os efeitos cromáticos, chamada tabela de cores. Essa tabela assume, em cada tela, um tamanho e uma posição diferentes.

Neste artigo, mostraremos a organização de duas outras tabelas da VRAM, existentes nos tipos de tela que permitem a programação de sprites: a tabela de atributos de sprites e a tabela de padrões de sprites. Mais adiante, trataremos da organização do VDP.

UM BANCO DE SPRITES

Antes de prosseguir no estudo da VRAM, convém recordar como o computador armazena o padrão de um spri-



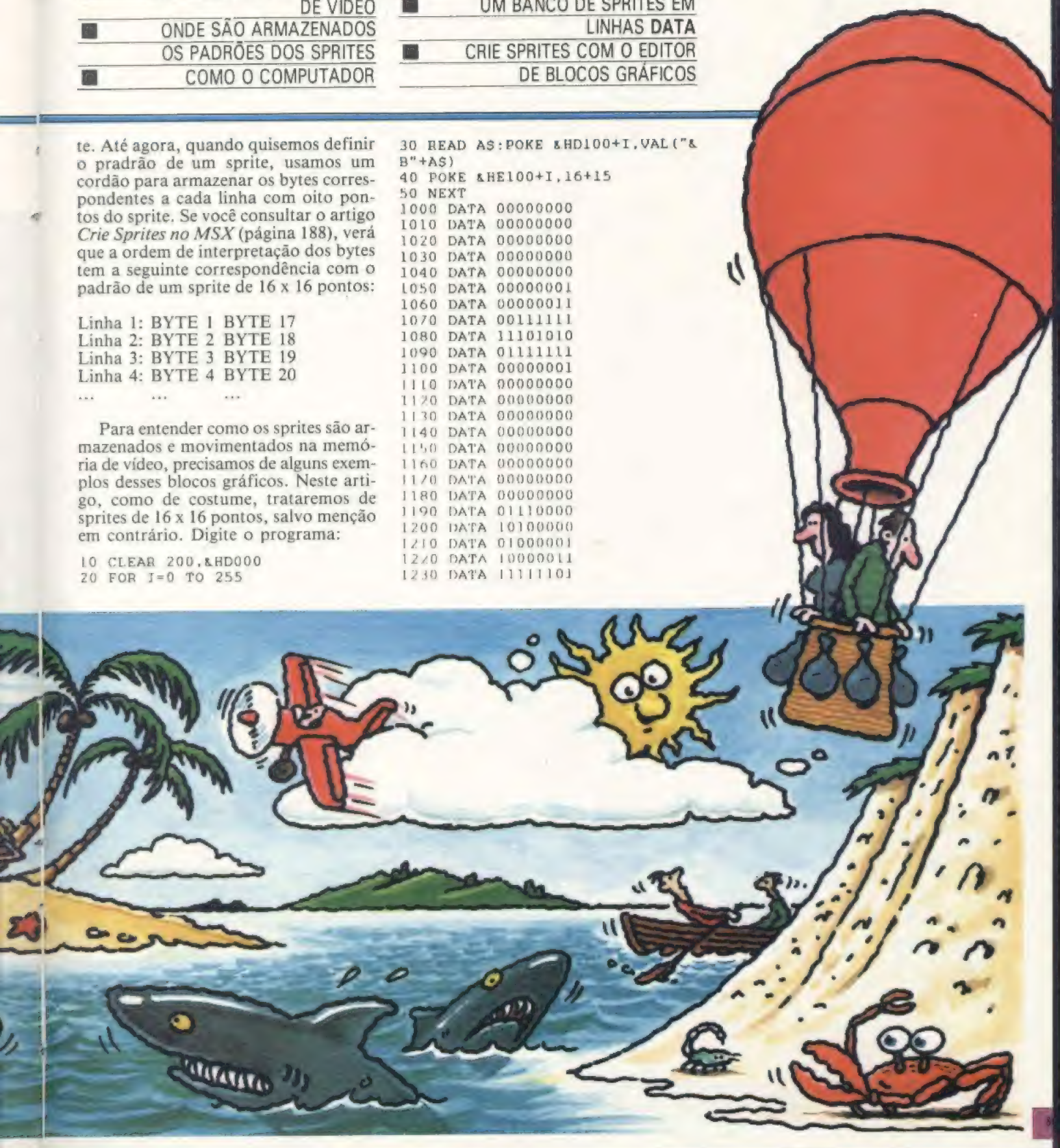
- ORGANIZAÇÃO DA MEMÓRIA DE VÍDEO
- ONDE SÃO ARMAZENADOS OS PADRÕES DOS SPRITES
- COMO O COMPUTADOR MOVIMENTA SPRITES
- UM BANCO DE SPRITES EM LINHAS **DATA**
- CRIE SPRITES COM O EDITOR DE BLOCOS GRÁFICOS

te. Até agora, quando quisemos definir o pradrão de um sprite, usamos um cordão para armazenar os bytes correspondentes a cada linha com oito pontos do sprite. Se você consultar o artigo *Crie Sprites no MSX* (página 188), verá que a ordem de interpretação dos bytes tem a seguinte correspondência com o padrão de um sprite de 16 x 16 pontos:

Linha 1: BYTE 1 BYTE 17
Linha 2: BYTE 2 BYTE 18
Linha 3: BYTE 3 BYTE 19
Linha 4: BYTE 4 BYTE 20
... ...

Para entender como os sprites são armazenados e movimentados na memória de vídeo, precisamos de alguns exemplos desses blocos gráficos. Neste artigo, como de costume, trataremos de sprites de 16 x 16 pontos, salvo menção em contrário. Digite o programa:

```
10 CLEAR 200,&HD000
20 FOR I=0 TO 255
30 READ A$:POKE &HD100+I,VAL("&
B"+A$)
40 POKE &HE100+I,16+15
50 NEXT
1000 DATA 00000000
1010 DATA 00000000
1020 DATA 00000000
1030 DATA 00000000
1040 DATA 00000000
1050 DATA 00000001
1060 DATA 00000011
1070 DATA 00111111
1080 DATA 11101010
1090 DATA 01111111
1100 DATA 00000001
1110 DATA 00000000
1120 DATA 00000000
1130 DATA 00000000
1140 DATA 00000000
1150 DATA 00000000
1160 DATA 00000000
1170 DATA 00000000
1180 DATA 00000000
1190 DATA 01110000
1200 DATA 10100000
1210 DATA 01000001
1220 DATA 10000011
1230 DATA 11111101
```



```
1240 DATA 10101001
1250 DATA 11111111
1260 DATA 00100000
1270 DATA 10010000
1280 DATA 01001000
1290 DATA 00111100
1300 DATA 00000000
1310 DATA 00000000
1320 DATA 00000000
1330 DATA 00000000
1340 DATA 00000000
1350 DATA 00111100
1360 DATA 00100010
1370 DATA 00100001
1380 DATA 01000001
1390 DATA 01000001
1400 DATA 10100010
1410 DATA 10000000
1420 DATA 11111101
1430 DATA 00110010
1440 DATA 00011101
1450 DATA 00000011
1460 DATA 00000000
1470 DATA 00000000
1480 DATA 00000000
1490 DATA 00000000
1500 DATA 00000000
1510 DATA 00000000
1520 DATA 00000000
1530 DATA 00000000
1540 DATA 11110000
1550 DATA 00101000
1560 DATA 00010100
1570 DATA 00101010
1580 DATA 01010101
1590 DATA 10111111
1600 DATA 01111000
1610 DATA 10000000
1620 DATA 00000000
1630 DATA 00000000
1640 DATA 00000000
1650 DATA 00000000
1660 DATA 00000000
1670 DATA 00000001
1680 DATA 00000001
1690 DATA 00000001
1700 DATA 11100001
1710 DATA 01010001
1720 DATA 00101010
1730 DATA 00100100
1740 DATA 00100000
1750 DATA 00101000
1760 DATA 01010100
1770 DATA 11100110
1780 DATA 00000101
1790 DATA 00000110
1800 DATA 00010010
1810 DATA 00000000
1820 DATA 00001001
1830 DATA 00000010
1840 DATA 10000000
1850 DATA 01000100
1860 DATA 00100001
1870 DATA 11111000
1880 DATA 00000100
1890 DATA 00110010
1900 DATA 00000001
1910 DATA 00111111
1920 DATA 01000000
1930 DATA 00111100
1940 DATA 11111000
1950 DATA 00000000
1960 DATA 00001110
1970 DATA 00000011
1980 DATA 00011011
1990 DATA 01101111
2000 DATA 10000111
2010 DATA 10011111
2020 DATA 00110011
2030 DATA 01100101
2040 DATA 01000101
2050 DATA 01000001
2060 DATA 00000001
2070 DATA 00000001
2080 DATA 00000001
2090 DATA 00000011
2100 DATA 00111111
2110 DATA 11111111
2120 DATA 00000000
2130 DATA 01111100
2140 DATA 11100110
2150 DATA 11111000
2160 DATA 11111100
2170 DATA 11100100
2180 DATA 11110010
2190 DATA 10011001
2200 DATA 10000100
2210 DATA 10000100
2220 DATA 10000000
2230 DATA 11000000
2240 DATA 11000000
2250 DATA 11000000
2260 DATA 11111100
2270 DATA 11111111
2280 DATA 00000111
2290 DATA 00111111
2300 DATA 00000000
2310 DATA 00000001
2320 DATA 00000010
2330 DATA 00000100
2340 DATA 00001000
2350 DATA 00010000
2360 DATA 00111111
2370 DATA 01111111
2380 DATA 00000000
2390 DATA 01111111
2400 DATA 00100000
2410 DATA 00011000
2420 DATA 00000111
2430 DATA 00000000
2440 DATA 10000000
2450 DATA 10000000
2460 DATA 10000000
2470 DATA 11000000
2480 DATA 11100000
2490 DATA 10110000
2500 DATA 10011000
2510 DATA 10000100
2520 DATA 10000010
2530 DATA 11111111
2540 DATA 10000000
2550 DATA 11111111
2560 DATA 00110010
2570 DATA 01100100
2580 DATA 11110000
2590 DATA 00000000
2600 DATA 00000000
2610 DATA 00000000
2620 DATA 00000000
2630 DATA 00000000
2640 DATA 00000000
2650 DATA 00000000
2660 DATA 00000000
2670 DATA 00000000
2680 DATA 00000000
2690 DATA 00000000
2700 DATA 00001100
2710 DATA 00011111
2720 DATA 00111111
2730 DATA 01111111
2740 DATA 11111111
2750 DATA 11111111
2760 DATA 00000000
```

```
2770 DATA 00000000
2780 DATA 00000000
2790 DATA 00000000
2800 DATA 00000000
2810 DATA 00000000
2820 DATA 00000000
2830 DATA 00010000
2840 DATA 00111000
2850 DATA 01111100
2860 DATA 11111100
2870 DATA 11111110
2880 DATA 11111110
2890 DATA 11111111
2900 DATA 11111111
2910 DATA 11111111
2920 DATA 00000000
2930 DATA 00000000
2940 DATA 00000000
2950 DATA 00000000
2960 DATA 00000011
2970 DATA 00000111
2980 DATA 00000110
2990 DATA 000C0010
3000 DATA 00000111
3010 DATA 00000111
3020 DATA 00000111
3030 DATA 11111111
3040 DATA 00111111
3050 DATA 00001111
3060 DATA 00000000
3070 DATA 00000000
3080 DATA 00000000
3090 DATA 00000000
3100 DATA 00000000
3110 DATA 00000000
3120 DATA 00000000
3130 DATA 10000000
3140 DATA 00000000
3150 DATA 00000000
3160 DATA 00000000
3170 DATA 11000000
3180 DATA 01000000
3190 DATA 10111111
3200 DATA 11011100
3210 DATA 11100000
3220 DATA 00001000
3230 DATA 00000100
3240 DATA 00000111
3250 DATA 00010001
3260 DATA 00111101
3270 DATA 01001101
3280 DATA 01101101
3290 DATA 01101101
3300 DATA 01101101
3310 DATA 00101101
3320 DATA 00011101
3330 DATA 00001101
3340 DATA 00000110
3350 DATA 00000001
3360 DATA 00000111
3370 DATA 00000100
3380 DATA 00000100
3390 DATA 00000011
3400 DATA 11000000
3410 DATA 00010000
3420 DATA 01111000
3430 DATA 01100100
3440 DATA 01101100
3450 DATA 01101100
3460 DATA 01101100
3470 DATA 01101000
3480 DATA 01110000
3490 DATA 01100000
3500 DATA 11000000
3510 DATA 00000000
3520 DATA 11000000
3530 DATA 01000000
3540 DATA 01000000
3550 DATA 10000000
```

Esse programa cria um banco de sprites no topo da memória. Os formatos das figuras podem ser observados nos números binários contidos nas linhas **DATA**; basta lembrar que os "1" correspondem a pontos "acesos" e os "0", a pontos "apagados" na tela. Note a ordem dos bytes que compõem o padrão do sprite. Ela é exatamente aquela a que fizemos referência: primeiro os dezesseis bytes da metade esquerda e depois os da metade direita.

A linha 10 protege o topo da memória, onde coloca os padrões. O endereço usado é o mesmo dos bancos de blocos criados pelo editor dos artigos *Geração de Blocos Gráficos* (1 e 2).

Os laços **FOR...NEXT** entre as linhas 20 e 50 usam o comando **READ** para obter os números binários das linhas **DATA** e **POKE** para colocar os valores correspondentes no topo da memória. A linha 40 cuida das cores dos padrões — preto sobre fundo branco — para que possamos vê-los através do editor.

Essa utilização do editor de blocos para observar os padrões criados não é obrigatória. O BASIC aqui apresentado foi criado com objetivos didáticos: ele permite compreender a organização da VRAM com o auxílio do programa do final do artigo. As figuras geradas por ele — barcos, peixes e uma ilha deserta, entre outros — serão utilizadas em outro artigo para animar um quadro na tela do computador.

USE O EDITOR DE BLOCOS GRÁFICOS

Há dois caminhos para utilizar o editor de blocos gráficos com esse conjunto de sprites. O primeiro é mais indicado para principiantes. Depois de executar e gravar o programa acima em fita, grave o banco de sprites por intermédio do comando:

```
BSAVE'CAS:SPRITE',&HD100,&HEFFF
```

Carregue então o programa editor de blocos gráficos do cassete e execute-o. Aperte a tecla T e responda L à pergunta do computador: "(S)AVE ou (L)OAD?". Posicione a fita na posição em que gravou o banco de sprites, aperte **ENTER** e, em seguida, pressione o botão "PLAY" do gravador.

O segundo caminho é mais indicado para aqueles que estão acostumados com o uso do editor: depois de executar o programa apresentado linhas atrás, carregue o editor de blocos da fita e execute-o também. (O banco de sprites está na posição correta, só que não podemos vê-lo. É possível, contudo, recu-

perar os padrões com auxílio da tecla R.) As posições ocupadas pelos blocos vão de 0 a 31.

Os blocos de 8 x 8 pontos que compõem os sprites estão ordenados de acordo com as exigências do VDP.

SPRITES NA VRAM

Nos tipos de tela que permitem o uso de sprites, certas porções da VRAM são reservadas para controlar sua exibição. Para exemplificar, recorreremos à tela de 32 colunas.

O MSX só permite a exibição de 32 sprites ao mesmo tempo. Cada um desses sprites tem suas características armazenadas numa tabela de atributos de sprites. São quatro os atributos de um sprite: suas coordenadas X e Y, seu nome e sua cor. Dessa forma, cada sprite necessita de quatro bytes para seus atributos, e a tabela de atributos terá um comprimento de 128 bytes.

A prioridade de um sprite é determinada pela sua posição na tabela de atributos. Terão prioridade os sprites que ocuparem as primeiras posições na tabela; ou seja, se dois sprites estiverem na mesma posição, será mostrado aquele que aparecer primeiro na tabela. As coordenadas X e Y determinam o lugar do sprite na tela. Elas correspondem ao canto superior esquerdo do sprite. Seus valores devem permanecer entre 0 e 255. Quando Y é maior que 191, o sprite desaparece na borda inferior da tela. Se Y for 208, todos os sprites com prioridade inferior desaparecerão; e, se for 209, o sprite em questão desaparecerá.

É importante prestar atenção aos limites impostos à coordenada X. Quando utilizamos a instrução **PUT SPRITE**, a coordenada X pode ter valores negativos. Contudo, se tentarmos colocar em uma posição da VRAM um valor fora da faixa que vai de 0 a 255, o computador emitirá uma mensagem de erro.

Infelizmente, um sprite só pode ter uma cor. Esta corresponderá aos "pontos acesos" do padrão, ou à cor de frente, que é determinada pelos quatro bits menos significativos do byte de cor, na tabela de atributos. Os códigos das cores são aqueles que você já conhece. O bit 7 dessa posição controla os valores negativos de X, quando utilizados em uma instrução **PUT SPRITE**, ou seja, ele será "aceso" se X estiver entre 0 e −32. Essas coordenadas negativas permitem que o sprite desapareça no canto esquerdo da tela, podendo reaparecer no direito.

O nome do sprite indica ao VDP onde o padrão do sprite pode ser encontrado na tabela de padrões de sprites.

O endereço inicial da tabela de atributos de sprites está em **BASE(8)** na tela de 32 colunas; em **BASE(13)** na tela de alta resolução; e em **BASE(18)** na tela multicolor.

O MSX permite que criemos até 64 sprites de 16 x 16 pontos, embora só 32 possam aparecer na tela ao mesmo tempo. Os padrões desses sprites ficam guardados na tabela de padrões. Como cada sprite precisa de 32 bytes para definir seu padrão, a tabela tem 2048 bytes de comprimento. Os padrões definidos pelo BASIC **SPRITE$(N)** são colocados em posições diferentes dessa tabela, conforme o valor de **N**. O VDP se baseia no nome do sprite que está na tabela de atributos para calcular a posição em que seu formato se encontra na tabela de padrões.

Cada sprite de 16 x 16 pontos necessita, ao ser programado, de 32 bytes. Se preferirmos trabalhar com sprites pequenos, com 8 x 8 pontos, cada um deles precisará de apenas oito bytes para armazenar seu padrão. Dessa forma, como a tabela de padrões de sprites tem 2048 bytes de comprimento, podemos criar até 256 sprites pequenos e até 64 sprites grandes.

O endereço inicial da tabela de padrões de sprites fica em **BASE(9)** na tela de 32 colunas; em **BASE(14)** na tela de alta resolução; e em **BASE(19)** na tela multicolor.

Quando seu micro padrão MSX estiver ligado em um modo gráfico que permita a exibição de sprites, o VDP procurará na tabela de atributos de sprites os dados necessários à exibição dos 32 sprites ali definidos. De um certo modo, podemos dizer que o VDP está *sempre* mostrando os 32 sprites na tela; nem sempre, contudo, vemos todos os sprites — seja porque o valor da coordenada Y está escondendo o sprite, seja porque não há nenhum padrão definido na tabela de padrões e todos os pontos dos sprites estão apagados.

UM PROGRAMA DIDÁTICO

Para que funcione o programa a seguir — organizado, como o anterior, com propósitos didáticos —, é necessária a presença de um banco de sprites no topo da memória. Portanto, antes de digitá-lo, execute o programa anterior, apague-o com **NEW** e só então digite e execute a nova listagem.

```
5 CLEAR 200,&HD000
10 SCREEN 1,2:KEY OFF
20 COLOR 1,7,7:CLS
30 Z$="00000000":N=0:W=0
40 A=BASE(8):GOSUB 3000
50 VPOKE BASE(6)+4,16+9
55 VPOKE BASE(6)+31,12*16+12
60 X=VPEEK(A+4*N+1):Y=VPEEK(A+4
*N):C=VPEEK(A+4*N+3):NX=VPEEK(A
+N*4+2):P=BASE(9)+8*NX
```

```
70 GOSUB 1000:GOSUB 2000
80 K$=INKEY$:IF K$="" THEN 80
90 IF K$="N" AND N<31 THEN N=N+
1:GOTO 60
100 IF K$="B" AND N>0 THEN N=N-
1:GOTO60
110 IF K$="C" THEN C=(C+1)MOD16
:VPOKE A+4*N+3,C:GOTO 60
120 IF K$=CHR$(28) AND X<247 TH
EN X=X+8:VPOKE A+4*N+1,X:GOTO 6
0
130 IF K$=CHR$(29) AND X>7 THEN
 X=X-8:VPOKE A+4*N+1,X:GOTO 60
140 IF K$=CHR$(31) AND Y<247 TH
EN Y=Y+8:VPOKE A+4*N,Y:GOTO 60
150 IF K$=CHR$(30) AND Y>7 THEN
 Y=Y-8:VPOKE A+4*N,Y:GOTO 60
160 IF K$="L" THEN VPOKE A+4*N,
160:GOTO 60
170 IF K$="A" THEN VPOKE A+4*N,
209:GOTO 60
180 IFK$=CHR$(27) THEN W=NOT W:
GOTO 60
190 IF K$="W" AND NX<252THEN NX
=NX+4:VPOKE A+4*N+2,NX:GOTO 60
200 IF K$="Q" AND NX>3 THEN NX=
NX-4:VPOKE A+4*N+2,NX:GOTO 60
210 GOTO 80
1000 FOR I=0 TO 9
1010 LOCATE 0,I+13
1020 PRINT RIGHT$(STR$(P+I),5);
" ";RIGHT$(Z$+BIN$(VPEEK(P+I)),
8);
1030 NEXT I:IF W=-1 THEN 1090
1040 FOR I=10 TO 31
1050 LOCATE 14,I-9
1060 PRINT P+I;RIGHT$(Z$+BIN$(V
PEEK(P+I)),8);
1070 NEXT I
1080 RETURN
1090 FOR I=10 TO 31
1100 LOCATE 14,I-9
1110 PRINT STRING$(7,32);RIGHT$
(Z$+BIN$(VPEEK(P+I)),8);
1120 IF I=30 THEN I=I+1:GOTO 10
50
1130 NEXT I:RETURN
2000 LOCATE 3,1
2010 PRINT "SPRITE";N;
2020 LOCATE 1,3
2030 PRINT A+4*N;VPEEK(A+4*N);T
AB(12);"Y"
2040 LOCATE 1,4
2050 PRINT A+4*N+1;VPEEK(A+4*N+
1);TAB(12);"X"
2060 LOCATE 1,5
2070 PRINT A+4*N+2;VPEEK(A+4*N+
2)
2080 LOCATE 1,6
2090 PRINT A+4*N+3;" ";HEX$(VPE
EK(A+4*N+3))
2100 LOCATE 4,8
2110 PRINT "BASE(8)";
2120 LOCATE 4,9
2130 PRINT BASE(8)
2140 LOCATE 4,10
2150 PRINT "BASE(9)";
2160 LOCATE 4,11
2170 PRINT BASE(9)
2180 FOR I=0 TO 12
2190 VPOKE BASE(5)+I*32+2,255
2200 VPOKE BASE(5)+I*32+15,255
2210 NEXT
2220 FOR I=3 TO 14
2230 VPOKE BASE(5)+I,255
2240 VPOKE BASE(5)+I+32*12,255
2250 NEXT:RETURN
3000 FOR I=0 TO 255
3010 VPOKE BASE(9)+I,PEEK(&HD10
0+I)
3020 NEXT:RETURN
```

Ao ser executado, o programa mostra, no canto superior esquerdo da tela, cercado por uma moldura verde, um quadro onde se lê "SPRITE 0". Esse quadro é uma imagem da tabela de atributos de sprites. Nele são exibidos o endereço e o conteúdo das quatro posições da VRAM que correspondem ao primeiro sprite da tabela — o sprite 0. Quanto menor o número do sprite, maior será sua prioridade, o que dá ao sprite zero prioridade absoluta. Para que o leitor se localize na VRAM, os endereços iniciais da tabela de atributos de sprites — **BASE(8)** — e da tabela de padrões de sprites — **BASE(9)** — também são mostrados no quadro.

Como já dissemos, o primeiro byte corresponde à coordenada Y. Se olharmos a tela, entenderemos por que o sprite 0 — o avião — não está aparecendo: a sua coordenada Y é maior que 191. Para fazê-lo aparecer, basta pressionar L. Feito isso, surge instantaneamente na tela; se observarmos novamente o quadro de moldura verde, vamos entender por quê: a coordenada Y do sprite 0 mudou para 160.

O sprite pode ser movimentado com o auxílio das teclas do cursor. Note como as coordenadas vão se modificando com o movimento do sprite — a coordenada X é o segundo byte do quadro.

O terceiro byte mostrado no interior da moldura verde corresponde ao "nome" do sprite. Esse valor orienta o VDP na busca do padrão do sprite dentro da tabela de padrões.

O valor do nome, multiplicado por oito e somado ao endereço inicial da tabela de padrões — **BASE(9)** —, fornece o endereço do primeiro dos 32 bytes correspondentes ao padrão do sprite. São exatamente estes 32 bytes que estão sendo mostrados no restante da tela. Ali temos uma imagem da porção da VRAM que contém o padrão do sprite. Os endereços dos bytes estão em decimal, e o seu conteúdo, em binário. Veja como os "zeros" e os "uns" desenham exatamente o perfil do avião. Confira também o endereço inicial, multiplicando o nome por oito e somando o resultado ao conteúdo de **BASE(9)**.

Voltando ao quadro de moldura verde, resta o quarto byte, que corresponde à cor do sprite, em hexadecimal. Podemos mudar instantaneamente a cor do sprite pressionando a tecla C. Ligue o sprite 0 com a tecla L, movimente-o com as setas e veja como sua cor muda com a tecla C. Observe atentamente o conteúdo dos bytes da tabela de atributos enquanto a cor e a posição do sprite vão sendo modificadas.

Para ver o próximo sprite na ordem de prioridade, aperte a tecla N. Os quatro bytes seguintes da tabela de atributos de sprites aparecerão dentro da moldura verde e os 32 bytes correspondentes ao desenho do sprite, junto com seus endereços na tabela de padrões de sprites, aparecerão no resto da tela. A tecla L faz o sprite aparecer, e a tecla A o apaga da tela, colocando 209 no byte da coordenada Y.

Podemos então percorrer a tabela de atributos de sprites pressionando repetidas vezes a tecla N. A região de interesse dentro da tabela de padrões de sprites também é atualizada. Podemos ligar, movimentar e mudar as cores dos sprites. Depois de usar a tecla N 31 vezes, chegaremos ao fim da tabela de atributos. Para voltar, tomando a direção dos sprites com maior prioridade (ou menor número), basta pressionar a tecla B, que tem um efeito inverso ao da tecla N.

Podemos ainda modificar o nome de um sprite, alterando a região da tabela de padrões a que ele se refere. Isso equivale a trocar o padrão do sprite. Para avançar na tabela de padrões, mantendo a prioridade do sprite, use a tecla W. Se o sprite em questão estiver na tela, veremos seu padrão ser substituído por outro desenho. Para retroceder na tabela, use Q.

As teclas N e B não modificam nenhum valor da VRAM; elas apenas mudam a região da tabela de atributos de sprites mostrada dentro da moldura verde. Já as teclas Q e W modificam o byte do nome do sprite, alterando a figura que está sendo mostrada. Por exemplo, a tecla W transforma o avião em uma nuvem, e a Q coloca um coqueiro no lugar onde havia um barco.

Use o programa para entender a organização da VRAM. Quando não houver mais dúvidas sobre os significados de cada endereço, podemos lançar mão da tecla ESC. Ela apaga a coluna de endereços da direita.

O programa ajuda também a ilustrar uma limitação do uso de sprites: não é possível inserir em uma mesma linha horizontal mais do que quatro sprites. Tente colocar cinco na mesma linha horizontal e veja o que acontece.

COMO FUNCIONA O PROGRAMA

A linha 5 mantém protegida a área onde foi colocado o banco de sprites. A 10 seleciona a tela de 32 colunas e sprites grandes. A 20 seleciona as cores da tela, além de limpá-la.

As linhas 30 e 40 estabelecem os valores iniciais de algumas variáveis. **Z$** serve para imprimir números binários, **N** é a posição do sprite na tabela de atributos — ou seja, sua prioridade —, **W** é um sinalizador que indica se ESC foi pressionada, **A** contém o endereço inicial da tabela.

A mesma linha 40 chama a sub-rotina 3000, que é responsável pela transferência do banco de sprites para a tabela de padrões. As linhas 50 e 55 modificam as cores de dois grupos de caracteres. Os caracteres 32 e 255 são usados para desenhar respectivamente as molduras laranja e verde.

O laço principal do programa começa na linha 60. Ali são atualizados os valores de uma série de variáveis. **X** é a coordenada X do sprite atual; **Y** é a outra coordenada desse sprite; **C** é a sua cor e **NX**, o seu nome. Todos esses valores são obtidos na tabela de atributos de sprites por meio de comandos **VPEEK**. **P** é o endereço inicial da porção da tabela de padrões onde se encontra o desenho do sprite atual; seu valor é calculado somando-se ao endereço inicial da tabela de padrões o resultado da multiplicação de oito pelo nome do sprite atual.

A seguir, a linha 70 chama duas sub-rotinas. A que começa na linha 1000 é responsável pela impressão da imagem da tabela de padrões na tela. A da linha 2000, por sua vez, cuida da impressão da imagem da tabela de atributos, junto com sua moldura verde.

A linha 80 aguarda que pressionemos uma tecla e a série de instruções **IF... THEN** que se segue executa as funções correspondentes a cada tecla.

Caso a tecla N tenha sido pressionada, a linha 90 modificará o sprite atual, passando ao sprite seguinte — de menor prioridade. Para fazer isso, basta aumentar o valor da variável **N** em uma unidade. A linha 100 cuida da tecla **B**, que volta ao sprite anterior, subtraindo uma unidade de **N**.

A linha 110 modifica a cor do sprite quando pressionamos a tecla C. A nova cor é obtida somando-se uma unidade à variável **C**. A função **MOD16** é utilizada para manter o código da cor entre 0 e 15. **C+1 MOD16** é o resto da divisão do código da nova cor por 16. O código da nova cor é colocado na tabela de atributos por **VPOKE**.

As linhas 120 e 130 cuidam do movimento horizontal do sprite. Conforme as teclas "seta para a esquerda" e "seta para direita" são presssionadas, o valor da variável **X** aumenta ou diminui oito unidades. Note que as condições **AND X<247** e **X>7** evitam que o sprite ultrapasse os limites da tela. A nova coordenada X é colocada na tabela de atributos por **VPOKE**.

As linhas 140 e 150 cuidam do movimento vertical de maneira análoga. A linha 160 "liga" o sprite quando L é precionada. Isso é feito com **VPOKE**, que coloca o valor 160 na posição da tabela de atributos que corresponde à coordenada Y. A linha 170 apaga o sprite ao toque da tecla A. Para obtermos esse efeito, a coordenada Y é modificada para o valor 209, com **VPOKE**. A linha 180 detecta a tecla ESC, modificando o valor do sinalizador **W**.

As linhas 190 e 200 alteram o valor do byte que corresponde ao nome do sprite atual. A tecla Q diminui o nome em quatro unidades e a tecla W o aumenta na mesma medida. O nome deve ser sempre múltiplo de 4 quando se trata de sprites grandes. É que um padrão de sprite tem 32 bytes. Como o nome é multiplicado por oito para calcular a posição do padrão na tabela, se o aumentarmos em quatro unidades, a posição na tabela avançará 8 x 4 = 32 unidades, passando para o padrão seguinte. Se o usuário pressionar qualquer outra tecla diferente das mencionadas, a linha 210 retornará à linha 80.

A sub-rotina da linha 1000 cuida da impressão dos valores contidos na tabela de padrões. O laço **FOR...NEXT** entre as linhas 1000 e 1030 imprime os dez primeiros endereços de interesse no canto inferior esquerdo da tela.

A seguir, conforme o valor de **W**, são impressos os outros 22 endereços. Se **W** for 0, o laço entre as linhas 1040 e 1070 imprimirá tanto os endereços quanto seus conteúdos. Se **W** for 1, o laço entre as linhas 1090 e 1130 imprimirá somente os conteúdos, preenchendo o espaço antes ocupado pelo endereço com espaços cor-de-laranja.

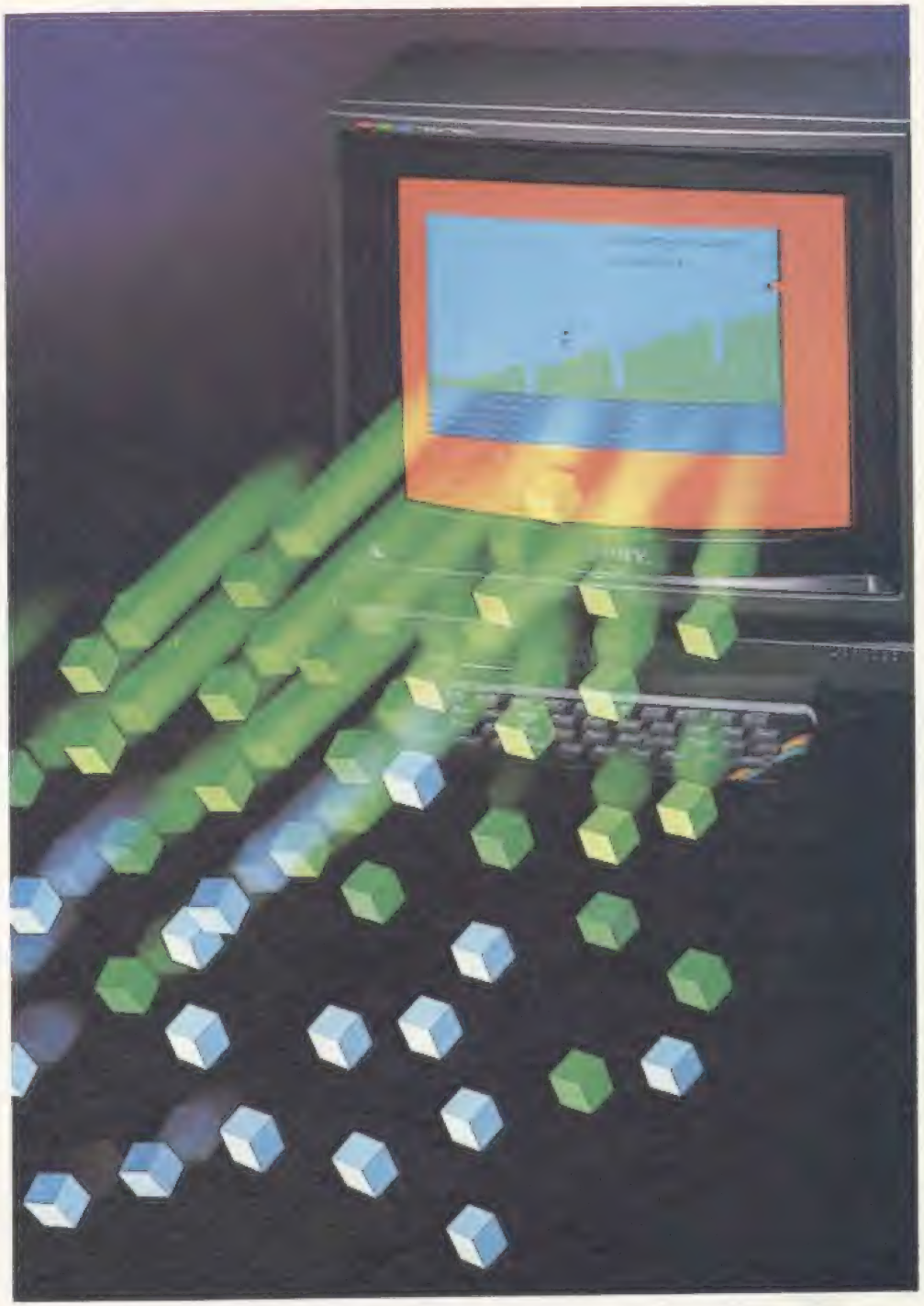
Os endereços são impressos com números decimais; os conteúdos aparecem em binário. A função **BIN$** calcula o valor binário; a função **RIGHT$**, juntamente com a variável **Z$**, faz com que sejam impressos exatamente oito dígitos binários por endereço. A função **STRING$** serve para imprimir sete espaços cor-de-laranja. A sub-rotina da linha 2000 cuida do quadro com moldura verde e do seu conteúdo. Os dois laços **FOR...NEXT** das linhas entre 2180 e 2250 desenham a moldura, que é composta por caracteres de código ASCII 255. No restante da rotina, **LOCATE** é utilizada juntamente com **PRINT** para escrever os endereços, conteúdos e suas legendas nas posições adequadas.

Os valores dos quatro bytes do sprinte atual são obtidos na tabela de atributos por meio do comando **VPEEK**.

BLOCOS GRÁFICOS EM AVALANCHE

- JUNTE AS PORÇÕES DO VIDEOGAME
- DEFINA A FORMA DOS BLOCOS GRÁFICOS
- NÃO GRAVE BYTES A MAIS

A procura do lanche roubado leva nosso indigitado herói a enfrentar novos obstáculos. Agora, ele precisa de uma montanha para escalar. Use blocos gráficos para construí-la.



Que não seja o Everest, nem tampouco o Aconcágua. Nossa montanha há de ser modesta e simples: um mero acidente geográfico. Mas que represente um obstáculo a transpor, um desafio a vencer. Que Willie, ao vê-la, não resista ao impulso de escalá-la.

Para colocá-la na tela, precisamos criar uma série de blocos gráficos. Mais uma vez, usaremos um programa BASIC para colocar na memória a tabela com os padrões dos blocos. Não será possível (por enquanto) ver os gráficos na tela de alta resolução — a montagem do cenário é tarefa para uma rotina em código que será abordada no próximo artigo. Os usuários do Spectrum e do MSX, contudo, poderão ter uma idéia dos blocos, olhando o padrão dos bits nas linhas **DATA**. Neste estágio, tudo o que você deve fazer é digitar e executar o programa BASIC e gravar os códigos resultantes em fita.

A esta altura, temos vários pedaços de videogame gravados separadamente. Chegou o momento de juntá-los.

S

Precisamos, em primeiro lugar, definir o formato dos blocos gráficos utilizados no jogo — nuvens, gaivotas, pedras que rolam, buracos, cobras venenosas, lanche para um piquenique na relva, e o próprio Willie, nosso desventurado herói. Só esses caracteres já dariam muitos bytes, mas, para suavizar o processo de animação, todas as figuras móveis serão desenhadas em várias posições, o que aumenta ainda mais os dados. Além disso, empregaremos saltos de meio caractere e alternaremos os blocos para dar a impressão de movimento contínuo.

```
 5 CLEAR 56999
10 FOR n=57000 TO 57327: READ
a: LET a$=STR$ a: POKE n,VAL
("BIN"+a$): NEXT n
9010 DATA 00011000
9011 DATA 00111100
9012 DATA 00111100
9013 DATA 00011000
9014 DATA 00111100
9015 DATA 00111100
9016 DATA 00111100
9017 DATA 00111100
9018 DATA 00111100
9019 DATA 00111100
9020 DATA 00011000
```

```
9021 DATA 00011000
9022 DATA 00011000
9023 DATA 00011000
9024 DATA 00011000
9025 DATA 00011110
9026 DATA 00000001
9027 DATA 00000011
9028 DATA 00000011
9029 DATA 00000001
9030 DATA 00000000
9031 DATA 00000001
9032 DATA 00000001
9033 DATA 00000001
9034 DATA 10000000
9035 DATA 11000000
9036 DATA 11000000
9037 DATA 10000000
9038 DATA 00000000
9039 DATA 00000000
9040 DATA 00000000
9041 DATA 11100000
9042 DATA 00001110
9043 DATA 00000000
9044 DATA 00000001
9045 DATA 00000010
9046 DATA 00000100
9047 DATA 00001000
9048 DATA 00000100
9049 DATA 00000000
9050 DATA 00000000
9051 DATA 00000000
9052 DATA 10000000
9053 DATA 01000000
9054 DATA 00100000
9055 DATA 00100000
9056 DATA 00110000
9057 DATA 00000000
9058 DATA 00000000
9059 DATA 00000000
9060 DATA 00000000
9061 DATA 00000000
9062 DATA 00011000
9063 DATA 00111100
9064 DATA 00111100
9065 DATA 00011000
9066 DATA 00000000
9067 DATA 00010000
9068 DATA 00010000
9069 DATA 00011110
9070 DATA 11100000
9071 DATA 00000000
9072 DATA 00001100
9073 DATA 00100100
9074 DATA 01000010
9075 DATA 10000010
9076 DATA 01000011
9077 DATA 00000000
9078 DATA 00000000
9079 DATA 00000000
9080 DATA 00000000
9081 DATA 00000000
9082 DATA 00000000
9083 DATA 00000000
9084 DATA 00000000
9085 DATA 00000000
9086 DATA 00000001
9087 DATA 00000011
9088 DATA 00000011
9089 DATA 00000001
9090 DATA 00000000
9091 DATA 00000000
9092 DATA 00000000
9093 DATA 00000000
9094 DATA 10000000
9095 DATA 11000000
9096 DATA 11000000
9097 DATA 10000000
9098 DATA 00000000
9099 DATA 00000001
9100 DATA 00000001
9101 DATA 00000001
9102 DATA 00001110
9103 DATA 00000000
9104 DATA 00000001
9105 DATA 00000010
9106 DATA 00000000
9107 DATA 00000000
9108 DATA 00000000
9109 DATA 11100000
9110 DATA 00000000
9111 DATA 00000000
9112 DATA 10000000
9113 DATA 01000000
9114 DATA 00000100
9115 DATA 00001000
9116 DATA 00000100
9117 DATA 00000000
9118 DATA 00000000
9119 DATA 00000000
9120 DATA 00000000
9121 DATA 00000000
9122 DATA 00100000
9123 DATA 00100000
9124 DATA 00110000
9125 DATA 00000000
9126 DATA 00000000
9127 DATA 00000000
9128 DATA 00000000
9129 DATA 00000000
9130 DATA 00011100
9131 DATA 00111110
9132 DATA 01111111
9133 DATA 11111111
9134 DATA 11111111
9135 DATA 11111110
9136 DATA 11111100
9137 DATA 00111000
9138 DATA 00000011
9139 DATA 00000111
9140 DATA 00001111
9141 DATA 00001111
9142 DATA 00001111
9143 DATA 00000111
9144 DATA 00000011
9145 DATA 00000001
9146 DATA 10000000
9147 DATA 11000000
9148 DATA 11100000
9149 DATA 11110000
9150 DATA 11110000
9151 DATA 11110000
9152 DATA 11100000
9153 DATA 11000000
9154 DATA 00000000
9155 DATA 00000000
9156 DATA 00000111
9157 DATA 00011000
9158 DATA 00100000
9159 DATA 01000000
9160 DATA 01000000
9161 DATA 10000000
9162 DATA 00000000
9163 DATA 00000000
9164 DATA 00011111
9165 DATA 10100000
9166 DATA 11000000
9167 DATA 00000000
9168 DATA 00000000
9169 DATA 00000000
9170 DATA 00000000
9171 DATA 00000000
9172 DATA 10000000
9173 DATA 01000000
9174 DATA 01011100
9175 DATA 00100010
9176 DATA 00000010
9177 DATA 00000010
9178 DATA 10000000
9179 DATA 01000000
9180 DATA 01111100
9181 DATA 00000010
9182 DATA 00000010
9183 DATA 00000001
9184 DATA 00000000
9185 DATA 00000000
9186 DATA 00000000
9187 DATA 00000000
9188 DATA 00000000
9189 DATA 00000100
9190 DATA 00001010
9191 DATA 00010001
9192 DATA 11100000
9193 DATA 00000000
9194 DATA 00000010
9195 DATA 00000100
9196 DATA 00001000
9197 DATA 00000100
9198 DATA 00000100
9199 DATA 00000100
9200 DATA 11111000
9201 DATA 00000000
9202 DATA 00000000
9203 DATA 00000000
9204 DATA 01111000
9205 DATA 10000110
9206 DATA 00000001
9207 DATA 00000001
9208 DATA 00000000
9209 DATA 00000000
9210 DATA 00000000
9211 DATA 00000000
9212 DATA 00011110
9213 DATA 01100001
9214 DATA 10000000
9215 DATA 10000000
9216 DATA 00000000
9217 DATA 00000000
9218 DATA 00000000
9219 DATA 00000000
9220 DATA 00000000
9221 DATA 00000000
9222 DATA 10000111
9223 DATA 01111001
9224 DATA 00000000
9225 DATA 00000000
9226 DATA 00000000
9227 DATA 00000000
9228 DATA 00000000
9229 DATA 00000000
9230 DATA 11100001
9231 DATA 10011110
9232 DATA 00000000
9233 DATA 00000000
9234 DATA 00100010
9235 DATA 00010100
9236 DATA 00001000
9237 DATA 00001000
9238 DATA 00001000
9239 DATA 00001000
```

```
9240 DATA 00001000
9241 DATA 00001000
9242 DATA 00011000
9243 DATA 00111100
9244 DATA 00110110
9245 DATA 01111110
9246 DATA 01111110
9247 DATA 00111100
9248 DATA 00011000
9249 DATA 00011000
9250 DATA 00011000
9251 DATA 00011000
9252 DATA 00001100
9253 DATA 00001100
9254 DATA 00000110
9255 DATA 00000110
9256 DATA 00000011
9257 DATA 00000011
9258 DATA 00000110
9259 DATA 00000110
9260 DATA 00001100
9261 DATA 00001100
9262 DATA 00011000
9263 DATA 00011000
9264 DATA 00110000
9265 DATA 00110000
9266 DATA 01100000
9267 DATA 01100000
9268 DATA 11000110
9269 DATA 11000011
9270 DATA 01100110
9271 DATA 01101100
9272 DATA 00111000
9273 DATA 00111000
9274 DATA 10000100
9275 DATA 11010110
9276 DATA 11111111
9277 DATA 11111111
9278 DATA 11111111
9279 DATA 11111111
9280 DATA 11111111
9281 DATA 11111111
9282 DATA 00000000
9283 DATA 00000001
9284 DATA 00000011
9285 DATA 00000111
9286 DATA 00011111
9287 DATA 00111111
9288 DATA 01111111
9289 DATA 11111111
9290 DATA 00000000
9291 DATA 00000000
9292 DATA 11111111
9293 DATA 11111111
9294 DATA 00111100
9295 DATA 00111100
9296 DATA 11111111
9297 DATA 11111111
9298 DATA 00000110
9299 DATA 00001000
9300 DATA 01110110
9301 DATA 11111111
9302 DATA 11111111
9303 DATA 11111111
9304 DATA 01111110
9305 DATA 00111100
9306 DATA 00010000
9307 DATA 00010000
9308 DATA 00010000
9309 DATA 00111000
9310 DATA 00111000
9311 DATA 00111000
9312 DATA 00111000
9313 DATA 00111000
9314 DATA 00010000
9315 DATA 00111000
9316 DATA 01111100
9317 DATA 00111000
9318 DATA 00111000
9319 DATA 00111000
9320 DATA 00010000
9321 DATA 00010000
9322 DATA 00000000
9323 DATA 00000000
9324 DATA 00000000
9325 DATA 00100000
9326 DATA 01010001
9327 DATA 10001010
9328 DATA 00000100
9329 DATA 00000000
9330 DATA 00000000
9331 DATA 00000000
9332 DATA 00000000
9333 DATA 10000010
9334 DATA 01000101
9335 DATA 00101000
9336 DATA 00010000
9337 DATA 00000000
```

Execute o programa e grave a tabela resultante com o comando:

```
SAVE 'AVAL4' CODE 57000,327
```

JUNTE OS PEDAÇOS

Longos programas em código geralmente têm de ser montados por partes. Isso é bom para testar as rotinas individualmente, mas tem a desvantagem de deixar para o leitor a tarefa de juntar os pedaços do programa. O fato de as rotinas isoladas serem executadas com sucesso não garante o funcionamento do programa completo.

A maior dificuldade surge quando uma das partes apaga outra durante a montagem. Se você gravou mais bytes do que devia junto com um dos segmentos do programa, pode acontecer que eles apaguem parte do outro segmento adjacente na memória. Quando gravarmos os códigos de uma porção do jogo, usando o monitor de *INPUT*, devemos informar o endereço inicial e o número exato de bytes a transferir para a fita. Se utilizarmos o comando **SAVE** do Spectrum devemos recorrer à sintaxe:

```
SAVE 'nome' CODE
```

seguida do endereço inicial, uma vírgula e o número de bytes a gravar.

Quando usamos o Assembler para montar os programas listados nos artigos, o endereço inicial é sempre igual à origem: o número que vem após o falso mnemônico **org**. O próprio Assembler cuida de começar a rotina nesse endereço. Nos programas BASIC que criam tabelas de dados na memória a partir de linhas **DATA** e usando **POKE**, o endereço inicial da tabela pode ser encontrado na própria listagem: valor inicial da variável de controle do laço **FOR... NEXT** responsável pela leitura e transferência dos dados para a memória.

Calcular o número exato de bytes a serem gravados já é mais difícil. O endereço final informado pelo Assembler nem sempre é o do final da rotina. Muitas vezes aquele endereço é apenas o da última instrução montada, que, em várias listagens, é um rótulo seguido por um asterisco, usado para cálculo de saltos e desvios, e não correspondendo ao final do programa.

Certos programadores gostam, por segurança, de gravar alguns bytes a mais; eles correm o risco, porém, de apagar pedaços de outras rotinas no processo de montagem do jogo completo.

A maneira mais segura de juntar as partes do programa é ler as rotinas no gravador, preenchendo a memória em ordem crescente. Porções com endereços iniciais menores devem ser recuperadas primeiro, sendo seguidas sempre das rotinas que ocupam as posições sucessivas. Isso fará os eventuais bytes gravados em excesso serem apagados pela próxima rotina (e não o contrário). Além disso, qualquer instrução **ret**, colocada no final de uma rotina apenas para permitir o seu funcionamento isolado, também será apagada.

Quando todas as rotinas montadas tiverem sido colocadas na memória e estiverem funcionando adequadamente, o programa resultante deve ser gravado inteiro, com um outro nome. Seu endereço inicial será o da primeira rotina lida na fita e o endereço final será o endereço final da última rotina colocada na memória. Qualquer problema pode exigir que as rotinas sejam reunidas novamente. Não se esqueça de gravar as listagens Assembly — os programas-fonte —, bem como os programas BASIC que criam as tabelas na memória, pois você pode precisar repetir o processo de montagem.

Se houver qualquer problema com as tabelas de dados, podemos montar as rotinas em código na ordem citada, e executar cada um dos programas BASIC. Eles devem ser lidos, um a um, no gravador, executados, e, em seguida, apagados com **NEW**. Depois disso, o próximo programa criador de tabelas será lido na fita e o processo se repetirá.

De início, é preciso definir o formato dos blocos gráficos utilizados no jogo — nuvens, gaivotas, pedras que rolam, buracos, cobras venenosas, gulo-

seimas para um piquenique, e o próprio Willie. No MSX é mais fácil movimentar figuras usando sprites para imprimir suavidade à animação.

As figuras definidas pelo programa a seguir têm a estrutura de sprites. Nem todas, porém, serão usadas como tal. O MSX só permite a exibição simultânea de 32 sprites; destes, apenas quatro, no máximo, podem ocupar a mesma linha horizontal. Mas isso poderia fazer o pobre Willie desaparecer da tela. (Imagine se houvesse quatro serpentes na mesma linha horizontal!)

```
10 CLEAR 200,-15200
20 FOR I=0 TO 639
30 READ A$:POKE -15200+I,VAL("&
B"+A$)
40 NEXT
9000 DATA 00011000
9010 DATA 00111100
9020 DATA 00111100
9030 DATA 00011000
9040 DATA 00111100
9050 DATA 00111100
9060 DATA 00111100
9070 DATA 00111100
9080 DATA 00111100
9090 DATA 00111100
9100 DATA 00011000
9110 DATA 00011000
9120 DATA 00011000
9130 DATA 00011000
9140 DATA 00011000
9150 DATA 00011110
9160 DATA 0,0,0,0,0,0,0,0
9170 DATA 0,0,0,0,0,0,0,0
9180 DATA 00000001
9190 DATA 00000011
9200 DATA 00000011
9210 DATA 00000001
9220 DATA 00000000
9230 DATA 00000001
9240 DATA 00000001
9250 DATA 00000001
9260 DATA 00001110
9270 DATA 00000000
9280 DATA 00000001
9290 DATA 00000010
9300 DATA 00000100
9310 DATA 00001000
9320 DATA 00000100
9330 DATA 00000000
9340 DATA 10000000
9350 DATA 11000000
9360 DATA 11000000
9370 DATA 10000000
9380 DATA 00000000
9390 DATA 00000000
9400 DATA 00000000
9410 DATA 11100000
9420 DATA 00000000
9430 DATA 00000000
9440 DATA 10000000
9450 DATA 01000000
9460 DATA 0010000
9470 DATA 00100000
9480 DATA 00110000
9490 DATA 00000000
9500 DATA 0,0,0,0,0,0,0,0
9510 DATA 0,0,0,0,0,0,0,0
9520 DATA 00011000
9530 DATA 00111100
9540 DATA 00111100
9550 DATA 00011000
9560 DATA 00000000
9570 DATA 00010000
9580 DATA 00010000
9590 DATA 00011110
9600 DATA 11100000
9610 DATA 00000000
9620 DATA 00001100
9630 DATA 00100100
9640 DATA 01000010
9650 DATA 10000010
9660 DATA 01000011
9670 DATA 00000000
9680 DATA 0,0,0,0,0,0,0,0
9690 DATA 00011100
9700 DATA 00111110
9710 DATA 01111111
9720 DATA 11111111
9730 DATA 11111111
9740 DATA 11111110
9750 DATA 11111100
9760 DATA 00111000
9770 DATA 0,0,0,0,0,0,0,0
9780 DATA 0,0,0,0,0,0,0,0
9790 DATA 0,0,0,0,0,0,0,0
9800 DATA 00000011
9810 DATA 00000111
9820 DATA 00001111
9830 DATA 00001111
9840 DATA 00001111
9850 DATA 00000111
9860 DATA 00000011
9870 DATA 00000001
9880 DATA 0,0,0,0,0,0,0,0
9890 DATA 10000000
9900 DATA 11000000
9910 DATA 11100000
9920 DATA 11110000
9930 DATA 11110000
9940 DATA 11110000
9950 DATA 11100000
9960 DATA 11000000
9970  DATA 00000000
9980  DATA 00000000
9990  DATA 00000111
10000 DATA 00011000
10010 DATA 00100000
10020 DATA 01000000
10030 DATA 01000000
10040 DATA 10000000
10050 DATA 10000000
I0060 DATA 01000000
10070 DATA 01111100
10080 DATA 00000010
10090 DATA 00000010
10100 DATA 00000001
10110 DATA 00000000
10120 DATA 00000000
10130 DATA 00000000
10140 DATA 00000000
10150 DATA 00011111
10160 DATA 10100000
10170 DATA 11000000
10180 DATA 00000000
10190 DATA 00000000
10200 DATA 00000000
10210 DATA 00000000
10220 DATA 00000000
10230 DATA 00000000
10240 DATA 00000100
10250 DATA 00001010
10260 DATA 00010001
10270 DATA 11100000
10280 DATA 00000000
10290 DATA 00000000
10300 DATA 00000000
10310 DATA 10000000
10320 DATA 01000000
10330 DATA 01011100
10340 DATA 00100010
10350 DATA 00000010
10360 DATA 00000010
10370 DATA 00000010
10380 DATA 00000100
10390 DATA 00001000
10400 DATA 00000100
10410 DATA 00000100
10420 DATA 00000100
10430 DATA 11111000
10440 DATA 00000000
10450 DATA 0,0,0,0,0,0,0,0
10460 DATA 0,0,0,0,0,0,0,0
10470 DATA 00000000
10480 DATA 00000000
10490 DATA 01111000
10500 DATA 10000110
10510 DATA 00000001
10520 DATA 00000001
10530 DATA 00000000
10540 DATA 00000000
10550 DATA 0,0,0,0,0,0,0,0
10560 DATA 00000000
10570 DATA 00000000
10580 DATA 00011110
10590 DATA 01100001
10600 DATA 10000000
10610 DATA 10000000
10620 DATA 00000000
10630 DATA 00000000
10640 DATA 0,0,0,0,0,0,0,0
10650 DATA 00000000
10660 DATA 00000000
10670 DATA 00000000
10680 DATA 00000000
10690 DATA 10000111
10700 DATA 01111001
10710 DATA 00000000
10720 DATA 00000000
10730 DATA 0,0,0,0,0,0,0,0
10740 DATA 00000000
10750 DATA 00000000
10760 DATA 00000000
10770 DATA 00000000
10780 DATA 11100001
10790 DATA 10011110
10800 DATA 00000000
10810 DATA 00000000
10820 DATA 0,0,0,0,0,0,0,0
10830 DATA 00100010
10840 DATA 00010100
10850 DATA 00001000
10860 DATA 00001000
10870 DATA 00001000
10880 DATA 00001000
10890 DATA 00001000
10900 DATA 00001000
10910 DATA 00011000
10920 DATA 00111100
10930 DATA 00110110
10940 DATA 01111110
10950 DATA 01111110
10960 DATA 00111100
10970 DATA 00011000
```

```
11120 DATA 00001100
11130 DATA 00011000
11140 DATA 00011000
11150 DATA 00110000
11160 DATA 00110000
11170 DATA 0,0,0,0,0,0,0,0
11180 DATA 0,0,0,0,0,0,0,0
11190 DATA 01100000
11200 DATA 01100000
11210 DATA 11000110
11220 DATA 11000011
11230 DATA 01100110
11240 DATA 01101100
11250 DATA 00111000
11260 DATA 00011000
11270 DATA 0,0,0,0,0,0,0,0
11280 DATA 0,0,0,0,0,0,0,0
11290 DATA 0,0,0,0,0,0,0,0
11300 DATA 10000100
11310 DATA 11010110
11320 DATA 11111111
11330 DATA 11111111
11340 DATA 11111111
11350 DATA 11111111
11360 DATA 11111111
11370 DATA 11111111
11380 DATA 0,0,0,0,0,0,0,0
11390 DATA 0,0,0,0,0,0,0,0
11400 DATA 0,0,0,0,0,0,0,0
11410 DATA 00000000
11420 DATA 00000001
11430 DATA 00000011
11440 DATA 00000111
11450 DATA 00011111
11460 DATA 00111111
11470 DATA 01111111
11480 DATA 11111111
11490 DATA 0,0,0,0,0,0,0,0
11500 DATA 0,0,0,0,0,0,0,0
11510 DATA 0,0,0,0,0,0,0,0
11520 DATA 00000000
11530 DATA 00000000
11540 DATA 11111111
11550 DATA 11111111
11560 DATA 00111100
11570 DATA 00111100
11580 DATA 11111111
11590 DATA 11111111
11600 DATA 0,0,0,0,0,0,0,0
11610 DATA 0,0,0,0,0,0,0,0
11620 DATA 0,0,0,0,0,0,0,0
11630 DATA 00000110
11640 DATA 00001000
11650 DATA 01110110
11660 DATA 11111111
11670 DATA 11111111
11680 DATA 11111111
11690 DATA 01111110
11700 DATA 00111100
10980 DATA 00011000
10990 DATA 0,0,0,0,0,0,0,0
11000 DATA 0,0,0,0,0,0,0,0
11010 DATA 00011000
11020 DATA 00011000
11030 DATA 00001100
11040 DATA 00001100
11050 DATA 00000110
11060 DATA 00000110
11070 DATA 00000011
11080 DATA 00000011
11090 DATA 00000110
11100 DATA 00000110
11110 DATA 00001100
11710 DATA 0,0,0,0,0,0,0,0
11720 DATA 0,0,0,0,0,0,0,0
11730 DATA 0,0,0,0,0,0,0,0
11740 DATA 00010000
11750 DATA 00010000
11760 DATA 00010000
11770 DATA 00111000
11780 DATA 00111000
11790 DATA 00111000
11800 DATA 00111000
11810 DATA 00111000
11820 DATA 0,0,0,0,0,0,0,0
11830 DATA 0,0,0,0,0,0,0,0
11840 DATA 0,0,0,0,0,0,0,0
11850 DATA 00010000
11860 DATA 00111000
11870 DATA 01111100
11880 DATA 00111000
11890 DATA 00111000
11900 DATA 00111000
11910 DATA 00010000
11920 DATA 00010000
11930 DATA 0,0,0,0,0,0,0,0
11940 DATA 0,0,0,0,0,0,0,0
11950 DATA 0,0,0,0,0,0,0,0
11960 DATA 00000000
11970 DATA 00000000
11980 DATA 00000000
11990 DATA 00100000
12000 DATA 01010001
12010 DATA 10001010
12020 DATA 00000100
12030 DATA 00000000
12040 DATA 0,0,0,0,0,0,0,0
12050 DATA 0,0,0,0,0,0,0,0
12060 DATA 0,0,0,0,0,0,0,0
12070 DATA 00000000
12080 DATA 00000000
12090 DATA 00000000
12100 DATA 10000010
12110 DATA 01000101
12120 DATA 00101000
12130 DATA 00010000
12140 DATA 00000000
12150 DATA 0,0,0,0,0,0,0,0
12160 DATA 0,0,0,0,0,0,0,0
12170 DATA 0,0,0,0,0,0,0,0
```

Execute o programa e grave a tabela resultante com o comando:

```
BSAVE 'CAS:AVAL' -15100,-14501
```

AS PARTES E O TODO

Longos programas em código devem ser montados por partes. Bom para testar as rotinas individualmente, esse método tem, contudo, a desvantagem de deixar para o leitor a tarefa de juntar os pedaços do programa. O fato de as rotinas isoladas serem executadas com sucesso não garante o funcionamento do programa completo.

Como no exemplo anterior, a maior dificuldade consiste em uma das partes ser apagada por outra durante o processo de montagem. Se você gravou mais bytes do que devia junto com um dos segmentos do programa, corre o risco de ver alguns desses bytes apagarem parte de outro pedaço, adjacente na memória. Assim, quando gravarmos os códigos de uma porção do jogo usando o monitor de *INPUT*, devemos informar o endereço inicial e o endereço final para determinar o número exato de bytes a serem transferidos para a fita. Se recorrermos ao comando **BSAVE** do MSX, devemos usar a sintaxe:

```
BSAVE 'CAS:NOME'
```

seguida do endereço inicial, uma vírgula e o endereço final da porção a ser gravada.

Quando usamos o Assembler para montar os programas dos artigos, o endereço inicial é sempre igual à origem (ou seja, o número que vem após o falso mnemônico **org**). O próprio Assembler cuida de começar a rotina nesse endereço. Nos programas BASIC que criam tabelas de dados na memória a partir de linhas **DATA** e usando **POKE**, o endereço inicial da tabela pode ser encontrado na própria listagem — valor inicial da variável de controle do laço **FOR...NEXT** responsável pela leitura e transferência dos dados para a memória.

Já o cálculo do número de bytes a gravar é mais difícil, pois o endereço final informado pelo Assembler nem sempre é o do final da rotina.

Há quem prefira gravar alguns bytes a mais, por segurança; nesse caso, porém, corre-se o risco de apagar pedaços de outras rotinas no processo de montagem do jogo completo.

O modo mais seguro de juntar as partes do programa consiste em ler as rotinas no gravador, preenchendo a memória em ordem crescente. Porções com endereços iniciais menores devem ser recuperadas da fita primeiro, sendo seguidas sempre das rotinas que ocupam posições sucessivas. Isso fará os bytes gravados em excesso serem apagados pela próxima rotina (e não o contrário). Assim, qualquer instrução **ret**, colocada no final de uma rotina apenas para permitir o seu funcionamento isolado, também será apagada.

Quando todas as rotinas montadas tiverem sido colocadas na memória e estiverem funcionando, o programa resultante deve ser gravado inteiro, com outro nome. Seu endereço inicial será o da primeira rotina lida na fita e o final será o endereço final da última rotina colocada na memória. Qualquer problema pode exigir que as rotinas sejam reunidas novamente. Não se esqueça de gravar as listagens Assembly — os programas-fonte — bem como os programas BASIC que criam as tabelas na memória, pois você pode precisar repetir o processo de montagem.

Se houver qualquer tipo de problema com as tabelas de dados, podemos montar as rotinas em código, na ordem citada, e executar cada um dos programas em BASIC. Eles devem ser lidos no gravador, um a um, postos em execução e, em seguida, apagados pelo comando **NEW**. Depois disso, o próximo programa criador de tabelas será lido na fita e o processo se repetirá.

Adicione as próximas linhas ao programa criador de tabelas que vem sendo montado ao longo destes artigos:

```
110 READ A$
120 FOR A=1 TO LEN (A$)
130 POKE AD,ASC(MID$(A$,A,1))
140 AD=AD+1
150 NEXT A
160 DATA ###!##!###!##!####!##!
##!###!##!!
170 FOR A=1 TO 702
180 READ A$:POKE AD,VAL("&H"+A$
)
190 AD=AD+1:NEXT A
200 IF AD<>18238 THEN PRINT "ER
ROR"
210 DATA 55,54,50,50,40,40,0,0,
7F,5F,57,D7,F5,FF,D5,75,DD,77,5
D,D5,7D,75,5D,77,F5,FD,57,75,DD
,77,5D,D5,FD,5F,57,D7,5D,FF,D5,
7F,75,77,FD,F7,D5,5D,75,77,55,D
5,D5,5D,5D,D7,F5,7D,D5,5D,5D,D7
,55,57,FF,7F,57,57,FD,FD
220 REM sol
230 DATA 55,75,D5,55,55,75,D5,5
5,75,75,D5,D5,5D,5D,D5,D5,5D,5D
,D7,55,57,5D,D7,5D,D5,D5,5D,5D,
75,D7,DD,75,7D,5D,75,D5,5D,75,5
D,D7,57,75,5D,5D,F5,D5,57,75,5D
,D5,57,75,57,55,55,D7,F7,55,55,
DD
240 DATA 57,55,55,D5,77,55,55,D
F,D5,D5,57,55,5D,D5,57,7F,F5,75
,5D,55,57,75,5D,F5,57,5D,75,5F,
5D,57,D7,55,75,75,57,55,55,77,7
5,D5,55,D7,75,D5,55,D7,75,75,57
,57,5D,75,57,57,5D,55,55,57,5D,
55
250 REM numeros
260 DATA 7D,D7,D7,D7,7D,5D,7D,5
D,5D,FF,7D,D7,5D,75,FF,FD,57,FD
,57,FD,5D,7D,DD,FF,5D,FF,D5,7D,
57,FD,7F,D5,FD,D7,7D,FF,57,5D,7
5,75,7D,D7,7D,D7,7D,7F,D7,7F,57
,57
270 REM graficos
280 DATA 57,5F,5F,57,5F,5F,5F,5
F,D5,F5,F5,D5,F5,F5,F5,F5,5F,5F
,57,57,57,57,57,57,F5,F5,D5,D5,
D5,D5,D5,FD,55,55,55,55,55,55,5
5,55,57,5F,5F,57,55,57,57,57,D5
,F5,F5,D5,55,55,55,FD,55,55,55,
55,55,55,55,55,55,55,55,55,55,5
5,55,55,FD,55,57,5D,75,D5,75,55
290 DATA 55,55,D5,75,5D,5D,5F,5
5,55,55,55,55,55,55,55,55,55,55
,55,55,57,5F,5F,57,55,55,55,55,
D5,F5,F5,D5,55,57,57,57,FD,55,5
7,5D,55,55,55,FD,55,55,D5,75,75
,D5,75,55,55,55,55,55,5D,5D,5F,
55,55,55,55,55,55,55,55,55,55,5
5,55,55,55,55,55,55,55,57,5F,5F,57
300 DATA 55,55,55,55,D5,F5,F5,D
5,55,55,55,55,55,55,55,55,55,55
,55,55,55,55,55,55,55,57,57,57,
FD,55,57,5D,55,55,55,FD,55,55,D
5,75,55,55,55,55,55,55,55,55,55
,55,55,55,55,55,55,55,75,D5,75,
55,55,55,55,55,55,5D,5D,5F,55,55,5
5,55,55,55,55
310 DATA 55,55,55,55,55,55,57,5
F,7F,FF,FF,FF,7F,5F,F5,FD,FF,FF
,FF,FD,F5,D5,55,55,55,55,55,55,
55,55,5F,7F,FF,FF,FF,7F,5F,57,D
5,F5,FD,FF,FF,FF,FD,F5,55,55,55
,55,55,55,55,55,5D,57,55,55,55,
55,55,55,5D,75,D5,D5,D5,D5,D5,D
5,57,5F,7D,7F,7F,5F,57,57,D5,F5
320 DATA FD,FD,FD,F5,D5,D5,57,5
7,55,55,55,55,55,55,D5,D5,F5,F5
,7D,7D,5F,5F,55,55,55,55,57,57,
5F,5F,7D,7D,F5,F5,D5,D5,55,55,7
D,7D,F5,F5,7D,7D,5F,5F,55,55,7D
,5F,7D,F5,D5,D5,55,55,AA,AA,56,
56,AA,AA,55,55,AA,AA,95,95,AA,A
A,55,55,7F,FF,FF,FF,7F,5F,81
330 DATA 15,7D,FF,FF,FF,FD,F5,5
7,57,57,5F,5F,5F,5F,5F,55,55,55
,D5,D5,D5,D5,57,5F,7F,5F,5F,5F,
57,57,55,D5,F5,D5,D5,D5,55,55,A
A,AA,AA,A6,99,6A,AA,AA,AA,AA,AA
,AA,A9,66,9A,AA,AA,AA,AA,6A,9A,
A6,A9,AA,AA,AA,AA,A6,99,6A,AA,A
A,
```

Os sustenidos e pontos de exclamação na linha 160 definem a silhueta da montanha. Cada sustenido corresponde a um caractere representativo de uma parte plana e cada ponto de exclamação, a um caractere de declive.

As linhas 210 a 330 contêm os bytes correspondentes aos padrões dos blocos gráficos utilizados — o sol, as pedras que rolam, as cobras venenosas e o próprio Willie. Pode parecer que há um excesso de dados na listagem, mas, para dar suavidade ao processo de animação, as figuras móveis serão desenhadas em posições diferentes.

Longos programas em código devem ser montados por partes. Isso é bom para testar as rotinas individualmente. Mas o sucesso na execução de rotinas isoladas não garante o funcionamento do programa completo. O principal perigo, neste como em outros casos, consiste em uma das partes apagar outra durante o processo de montagem. Assim, se você gravou mais bytes do que devia junto com um dos segmentos do programa, eles podem suprimir parte do outro pedaço adjacente na memória.

Quando gravarmos os códigos de uma porção do jogo usando o monitor de *INPUT*, devemos informar os endereços inicial e final, de modo a transferir para a fita o número exato de bytes. Se usarmos o comando **CSAVEM** do TRS-Color, devemos recorrer à sintaxe:

```
CSAVEM 'NOME',
```

seguida do endereço inicial, uma vírgula e o endereço final da porção a gravar. Quando usarmos o Assembler para montar os programas listados nos artigos, o endereço inicial será igual à origem (ou seja, o número que vem após o falso mnemônico **ORG**). O próprio Assembler cuidará de começar a rotina nesse endereço. Nos programas BASIC que criam tabelas de dados na memória a partir de linhas **DATA** e usando **POKE**, o endereço inicial da tabela pode ser encontrado na própria listagem — valor inicial da variável de controle do laço **FOR...NEXT** responsável pela leitura e transferência dos dados para a memória.

Mais difícil é a tarefa de calcular o número de bytes a serem gravados. É que o endereço final informado pelo Assembler nem sempre é o do final da rotina. Muitas vezes, aquele endereço é apenas o da última instrução montada, o que em muitas das nossas listagens significa um rótulo seguido por um asterisco, usado para cálculo de saltos e desvios, não correspondendo ao final do programa.

Procure não gravar bytes a mais para não apagar pedaços de outras rotinas no processo de montagem do jogo. A maneira mais segura de juntar as partes do programa consiste em ler as rotinas no gravador, preenchendo a memória em ordem crescente. Porções com endereços iniciais menores devem ser recuperadas da fita em primeiro lugar, sendo seguidas sempre das rotinas que ocupam posições sucessivas. Isso fará os bytes gravados em excesso serem apagados pela próxima rotina (e não o contrário). Assim, qualquer instrução **RTS**, colocada no final de uma rotina para permitir o seu funcionamento isolado, também será apagada.

Quando todas as rotinas montadas tiverem sido colocadas na memória e estiverem funcionando adequadamente, chegou o momento de gravar, inteiro, o programa resultante, com um outro nome. Seu endereço inicial será o da primeira rotina lida na fita e o endereço final será o endereço final da última rotina colocada na memória. Qualquer problema pode exigir que as rotinas sejam agrupadas novamente. Não se esqueça de gravas as listagens Assembly — os programas-fonte —, assim como os programas BASIC que criam as tabelas na memória, pois você pode precisar repetir o processo de montagem.

CLASSIFICAÇÃO DOS MICROCOMPUTADORES POR LINHA

LINHA	FABRICANTE	MODELO
Apple II +	Appletronica	Thor 2010
Apple II +	CCE	MC-4000 Exato
Apple II +	CPA	Absolutus
Apple II +	CPA	Polaris
Apple II +	Digitus	DGT-AP
Apple II +	Dismac	D-8100
Apple II +	ENIAC	ENIAC II
Apple II +	Franklin	Franklin
Apple II +	Houston	Houston AP
Apple II +	Magnex	DM II
Apple II +	Maxitronica	MX-2001
Apple II +	Maxitronica	MX-48
Apple II +	Maxitronica	MX-64
Apple II +	Maxitronica	Maxitronic I
Apple II +	Microcraft	Craf II Plus
Apple II +	Milmar	Apple II Plus
Apple II +	Milmar	Apple Master
Apple II +	Milmar	Apple Senior
Apple II +	Omega	MC-400
Apple II +	Polymax	Maxxi
Apple II +	Polymax	Poly Plus
Apple II +	Spectrum	Microengenho I
Apple II +	Spectrum	Spectrum ed
Apple II +	Suporte	Venus II
Apple II +	Sycomig	SIC I
Apple II +	Unitron	AP II
Apple II +	Victor do Brasil	Elppa II Plus
Apple II +	Victor do Brasil	Elppa Jr.
Apple IIe	Microcraft	Craft IIe
Apple IIe	Microdigital	TK-3000 IIe
Apple IIe	Spectrum	Microengenho II
MSX	Gradiente	Expert GPC-1
MSX	Sharp	Hotbit HB-8000
Sinclair Spectrum	Microdigital	TK-90X
Sinclair Spectrum	Timex	Timex 2000
Sinclair ZX-81	Apply	Apply 300
Sinclair ZX-81	Engebras	AS-1000
Sinclair ZX-81	Filcres	NEZ-8000
Sinclair ZX-81	Microdigital	TK-82C
Sinclair ZX-81	Microdigital	TK-83
Sinclair ZX-81	Microdigital	TK-85
Sinclair ZX-81	Prologica	CP-200
Sinclair ZX-81	Ritas	Ringo R-470
Sinclair ZX-81	Timex	Timex 1000
Sinclair ZX-81	Timex	Timex 1500
TRS-80 Mod. I	Dismac	D-8000
TRS-80 Mod. I	Dismac	D-8001/2
TRS-80 Mod. I	LNW	LNW-80
TRS-80 Mod. I	Video Genie	Video Genie I
TRS-80 Mod.III	Digitus	DGT-100
TRS-80 Mod.III	Digitus	DGT-1000
TRS-80 Mod.III	Kemitron	Naja 800
TRS-80 Mod.III	Prologica	CP-300
TRS-80 Mod.III	Prologica	CP-500
TRS-80 Mod.III	Sysdata	Sysdata III
TRS-80 Mod.III	Sysdata	Sysdata Jr.
TRS-80 Mod.IV	Multix	MX-Compacto
TRS-80 Mod.IV	Sysdata	Sysdata IV
TRS-Color	Codimex	CS-6508
TRS-Color	Dynacom	MX-1600
TRS-Color	LZ	Color 64
TRS-Color	Microdigital	TKS-800
TRS-Color	Prologica	CP-400

CLASSIFICAÇÃO DOS MICROCOMPUTADORES POR NOME

FABRICANTE	MODELO	PAÍS	LINHA
Appletronica	Thor 2010	Brasil	Apple II +
Apply	Apply 300	Brasil	Sinclair ZX-81
CCE	MC-4000 Exato	Brasil	Apple II +
CPA	Absolutus	Brasil	Apple II +
CPA	Polaris	Brasil	Apple II +
Codimex	CS-6508	Brasil	TRS-Color
Digitus	DGT-100	Brasil	TRS-80 Mod.III
Digitus	DGT-1000	Brasil	TRS-80 Mod.III
Digitus	DGT-AP	Brasil	Apple II +
Dismac	D-8000	Brasil	TRS-80 Mod. I
Dismac	D-8001/2	Brasil	TRS-80 Mod. I
Dismac	D-8100	Brasil	Apple II +
Dynacom	MX-1600	Brasil	TRS-Color
ENIAC	ENIAC II	Brasil	Apple II +
Engebras	AS-1000	Brasil	Sinclair ZX-81
Filcres	NEZ-8000	Brasil	Sinclair ZX-81
Franklin	Franklin	USA	Apple II +
Gradiente	Expert GPC1	Brasil	MSX
Houston	Houston AP	Brasil	Apple II +
Kemitron	Naja 800	Brasil	TRS-80 Mod.III
LNW	LNW-80	USA	TRS-80 Mod. I
LZ	Color 64	Brasil	TRS-Color
Magnex	DM II	Brasil	Apple II +
Maxitronica	MX-2001	Brasil	Apple II +
Maxitronica	MX-48	Brasil	Apple II +
Maxitronica	MX-64	Brasil	Apple II +
Maxitronica	Maxitronic I	Brasil	Apple II +
Microcraft	Craft II Plus	Brasil	Apple II +
Microcraft	Caft IIe	Brasil	Apple IIe
Microdigital	TK-3000 IIe	Brasil	Apple IIe
Microdigital	TK-82C	Brasil	Sinclair ZX-81
Microdigital	TK-83	Brasil	Sinclair ZX-81
Microdigital	TK-85	Brasil	Sinclair ZX-81
Microdigital	TK-90X	Brasil	Sinclair Spectrum
Microdigital	TKS-800	Brasil	TRS-Color
Milmar	Apple II Plus	Brasil	Apple II +
Milmar	Apple Master	Brasil	Apple II +
Milmar	Apple Senior	Brasil	Apple II +
Multix	MX-Compacto	Brasil	TRS-80 Mod.IV
Omega	MC-400	Brasil	Apple II +
Polymax	Maxxi	Brasil	Apple II +
Polymax	Poly Plus	Brasil	Apple II +
Prologica	CP-200	Brasil	Sinclair ZX-81
Prologica	CP-300	Brasil	TRS-80 Mod.III
Prologica	CP-400	Brasil	TRS-Color
Prologica	CP-500	Brasil	TRS-80 Mod.III
Ritas	Ringo R-470	Brasil	Sinclair ZX-81
Sharp	Hotbit HB-8000	Brasil	MSX
Spectrum	Microengenho I	Brasil	Apple II +
Spectrum	Microengenho II	Brasil	Apple IIe
Spectrum	Spectrum ed	Brasil	Apple II +
Suporte	Venus II	Brasil	Apple II +
Sycomig	SIC I	Brasil	Apple II +
Sysdata	Sysdata III	Brasil	TRS-80 Mod.III
Sysdata	Sysdata IV	Brasil	TRS-80 Mod.IV
Sysdata	Sysdata Jr.	Brasil	TRS-80 Mod.III
Timex	Timex 1000	USA	Sinclair ZX-81
Timex	Timex 1500	USA	Sinclair ZX-81
Timex	Timex 2000	USA	Sinclair Spectrum
Unitron	AP II	Brasil	Apple II +
Victor do Brasil	Elppa II Plus	Brasil	Apple II +
Victor do Brasil	Elppa Jr.	Brasil	Apple II +
Video Genie	Video Genie I	USA	TRS-80 Mod. I

UM LOGOTIPO PARA CADA MODELO DE COMPUTADOR

INPUT foi especialmente projetado para microcomputadores compatíveis com as sete principais linhas existentes no mercado.
Os blocos de textos e listagens de programas aplicados apenas a determinadas linhas de micros podem ser identificados por meio dos seguintes símbolos:

Sinclair ZX-81

TRS-80

TK-2000

MSX

Spectrum

TRS-Color

Apple II

Quando o emblema for seguido de uma faixa, então tanto o texto como os programas que se seguem passam a ser específicos para a linha indicada.

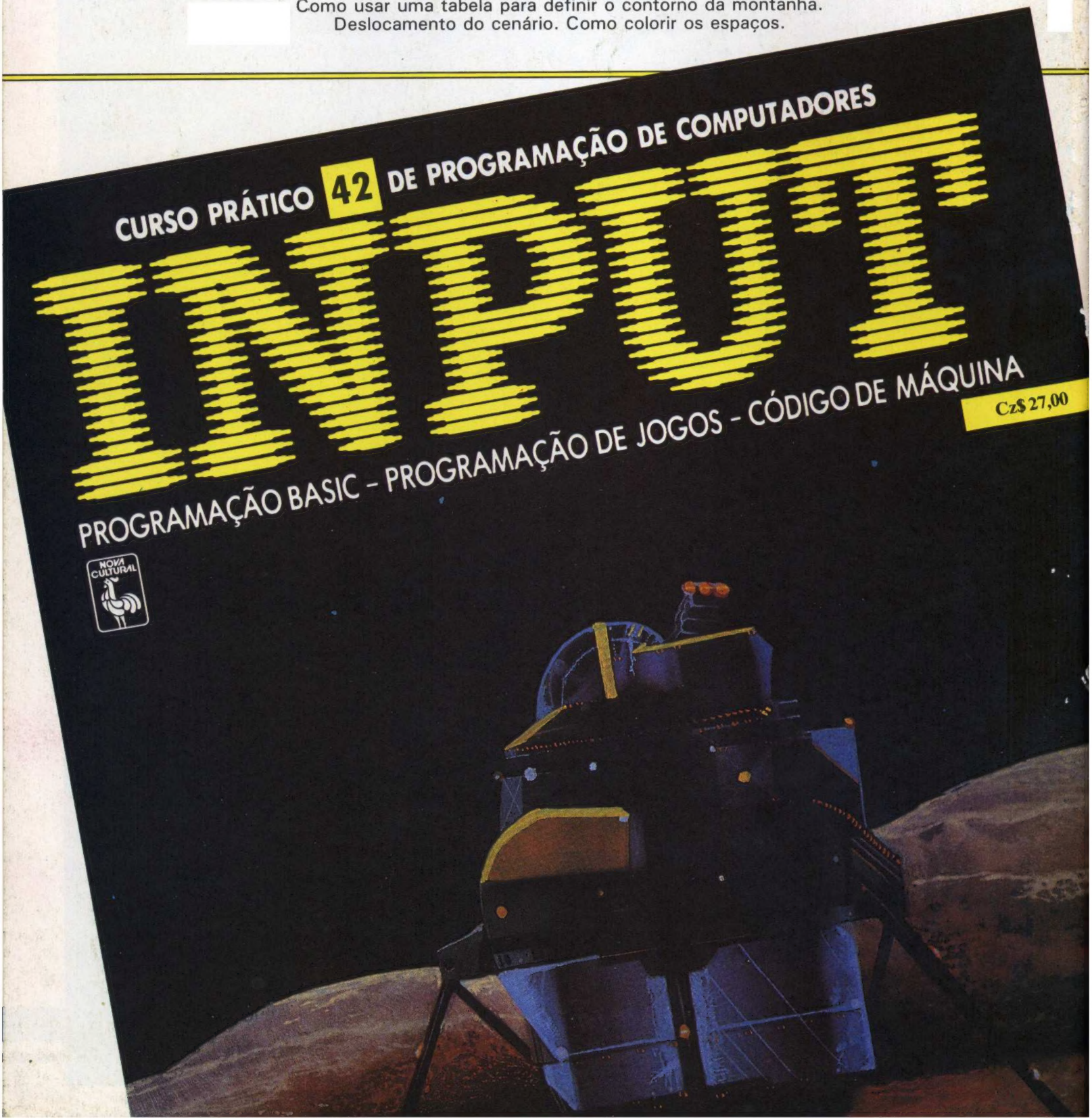
NO PRÓXIMO NÚMERO

APLICAÇÕES

Calendários mensais e anuais. Registros de eventos. Veja como o computador pode ajudá-lo a cumprir os compromissos.

PROGRAMAÇÃO DE JOGOS

Faça um pouso suave e tranqüilo na superfície da lua. Além de habilidade, você vai precisar de muito sangue frio.

CÓDIGO DE MÁQUINA

Como usar uma tabela para definir o contorno da montanha. Deslocamento do cenário. Como colorir os espaços.